



Guida per gli sviluppatori

Amazon Simple Storage Service



Versione API 2006-03-01

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Simple Storage Service: Guida per gli sviluppatori

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

I marchi e il trade dress di Amazon non possono essere utilizzati in relazione ad alcun prodotto o servizio che non sia di Amazon, in alcun modo che possa causare confusione tra i clienti, né in alcun modo che possa denigrare o screditare Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà dei rispettivi proprietari, che possono o meno essere affiliati, collegati o sponsorizzati da Amazon.

Table of Contents

Che cos'è Amazon S3?	1
Consigli per le chiamate API	2
Effettuare chiamate dirette all'API REST	3
Permissions	3
Sviluppo con Amazon S3	5
Esecuzione di richieste	5
Le chiavi di accesso	6
Endpoint della richiesta	8
Esecuzione di richieste su IPv6	8
Effettuare richieste utilizzando AWS SDK	18
Esecuzione di richieste con l'utilizzo di API REST	57
Utilizzando il AWS CLI	70
Scopri di più su AWS CLI	71
Sviluppo con SDK AWS	71
Interfacce di programmazione SDK	72
Specificare la versione della firma nell'autenticazione della richiesta	72
Ottieni gli ID di richiesta di Amazon S3 per Supporto	84
Utilizzo di AWS CLI per ottenere gli ID delle richieste	84
Utilizzo di Windows PowerShell per ottenere gli ID delle richieste	85
Utilizzo AWS CloudTrail eventi relativi ai dati per ottenere gli ID delle richieste	85
Utilizzo della registrazione degli accessi al server S3 per recuperare gli ID richiesta	85
Utilizzo di HTTP per recuperare gli ID richiesta	85
Utilizzo di un browser Web per recuperare gli ID richiesta	86
Utilizzo di AWS SDK per ottenere gli ID delle richieste	86
Operazioni API S3 a livello di oggetto supportate per S3 Tables	91
Esempi di codice	97
Simple Storage Service (Amazon S3)	103
Nozioni di base	108
Scenari	802
Esempi serverless	1257
Amazon S3 Control	1269
Nozioni di base	1270
Bucket di directory S3	1350
Nozioni di base	1352

Scenari	1477
Autenticazione delle richieste (AWS Signature (versione 4))	1481
Metodi di autenticazione	1482
Introduzione alla firma delle richieste	1483
Utilizzo di un'intestazione di autorizzazione	1484
Panoramica di	1485
Calcolo della firma: trasferimento del payload in un unico blocco	1490
Calcolo della firma: trasferimento del payload in più blocchi	1510
Calcolo della firma: incluse le intestazioni finali	1523
Utilizzo dei parametri di interrogazione	1530
Calcolo di una firma	1533
Un esempio	1538
Esempio 2	1540
Esempi: Calcoli delle firme	1540
Esempi di calcolo della firma utilizzando Java	1541
Esempi di calcolo della firma utilizzando C#	1542
Autenticazione delle richieste HTTP POST	1543
Calcolo di una firma	1546
Chiavi specifiche delle policy di autenticazione di Amazon S3 Signature versione 4	1547
Esempi di policy Bucket che utilizzano le chiavi di condizione relative alla versione 4 di Signature	1550
Browser-Based Caricamenti tramite POST	1554
POST Object	1555
Description	1555
Controllo delle versioni	1556
Richieste	1556
Esempi	1576
Risorse correlate	1577
POST Object restore	1578
Description	1578
Interrogazione degli archivi con Select Requests	1578
Ripristino degli archivi	1580
Richieste	1581
Risposte	1596
Esempi	1598
Ulteriori informazioni	1600

Browser-Based Caricamenti tramite HTTP POST	1600
Calcolo di una firma	1602
Creazione di moduli HTML	1603
Dichiarazione del modulo HTML	1604
Campi del modulo HTML	1605
Politica POST	1611
Scadenza	1612
Corrispondenza delle condizioni	1613
Condizioni	1615
Utilizzo di escape con caratteri	1619
Esempio di caricamento POST	1621
Caricamento di un file su Amazon S3 tramite HTTP POST	1621
Browser-based caricamenti tramite Amplify AWS	1624
Utilizzo di AWS JavaScript Libreria Amplify per caricare file su Amazon S3	1624
Ulteriori informazioni	1625
Intestazioni di richiesta comuni	1626
Intestazioni di risposta comuni	1631
Risposte agli errori	1636
Risposte agli errori REST	1637
Elenco dei codici di errore	1638
Elenco dei codici di errore SELECT Object Content	1668
Elenco dei codici di errore Replication-related	1682
Elenco dei codici di errore Tagging-related	1685
Elenco dei codici di errore relativi alle annotazioni di Amazon S3	1686
Elenco dei codici di errore di Amazon S3 on Outposts	1690
Elenco dei codici di errore di Amazon S3 Storage Lens	1691
Elenco dei codici di errore di Amazon S3 Object Lambda	1701
Elenco dei codici di errore asincroni di Amazon S3	1705
Elenco dei codici di errore di Amazon S3 Access Grants	1707
Elenco dei codici di FSx-related errore Amazon	1710
Elenco dei codici di errore di Amazon S3 Tables	1711
Best Practice per gli errori Amazon S3	1712
Riprova InternalErrors	1712
Ottimizza l'applicazione in caso di errori ripetuti SlowDown	1712
Errori isolati	1713
AWS Glossario	1714

Resources	1715
Appendice	1717
Appendice: Risposta SelectObjectContent	1718
Description	1718
Risposte	1718
Risorse correlate	1728
Appendice: oggetto OPTIONS	1730
Description	1730
Richieste	1730
Risposte	1731
Esempi	1733
Risorse correlate	1733
Appendice: API SOAP	1734
Operazioni sul servizio (API SOAP)	1734
Operazioni sui bucket (API SOAP)	1736
Operazioni sugli oggetti (API SOAP)	1749
Autenticazione delle richieste SOAP	1774
Impostazione della policy d'accesso con SOAP	1775
Elementi comuni	1777
Risposte agli errori SOAP	1777
Appendice: Autenticazione delle richieste (AWS firma (versione 2))	1779
Autenticazione delle richieste tramite l'API REST (AWS firma (versione 2))	1781
Firma e autenticazione delle richieste REST (AWS firma (versione 2))	1784
Browser-based caricamenti tramite POST (versione di AWS firma 2)	1799
Appendice: API di configurazione del ciclo di vita (obsoleto)	1823
Ciclo di vita del bucket PUT (obsoleto)	1825
Ciclo di vita del bucket GET (obsoleto)	1842
Cronologia dei documenti	1854
.....	mdccclv

Che cos'è Amazon S3?

Amazon Simple Storage Service (Amazon S3) è un servizio di archiviazione di oggetti che offre scalabilità, disponibilità dei dati, sicurezza e prestazioni tra le migliori del settore. I clienti di tutte le dimensioni e settori possono utilizzare Amazon S3 per archiviare e proteggere qualsiasi quantità di dati in un'ampia gamma di casi d'uso, come data lake, siti Web, applicazioni mobili, backup e ripristino, archivi, applicazioni per aziende, dispositivi IoT e analisi dei Big Data. Amazon S3 offre caratteristiche di gestione che consentono di ottimizzare, organizzare e configurare l'accesso ai dati per soddisfare specifici requisiti aziendali, organizzativi e di conformità.

Questa guida fornisce informazioni incentrate sugli sviluppatori sull'utilizzo dell'API REST di Amazon S3 per creare applicazioni che archiviano e recuperano dati nel cloud.

Per utilizzare l'API REST, è possibile servirsi di qualunque kit di strumenti in grado di supportare HTTP. Puoi anche utilizzare un browser per recuperare gli oggetti, purché siano leggibili in modo anonimo.

Poiché l'API REST utilizza codici di stato e intestazioni HTTP standard, i kit di strumenti e i browser standard funzionano come previsto. In alcune aree, abbiamo aggiunto funzionalità a HTTP (ad esempio, abbiamo aggiunto intestazioni per supportare il controllo degli accessi). In questi casi, abbiamo aggiunto la nuova funzionalità in modo che corrisponda allo stile di utilizzo standard di HTTP.

Note

Il supporto per SOAP su HTTP è obsoleto, ma è ancora disponibile su HTTPS. Tuttavia, le nuove funzionalità di Amazon S3 non saranno supportate per SOAP. Ti consigliamo di utilizzare questa API REST o gli AWS SDK al seguente link:

<https://aws.amazon.com/developer/tools/>

Questa guida per sviluppatori include:

- [Intestazioni di richiesta comuni](#) e [Intestazioni di risposta comuni](#)
- [Risposte agli errori](#)
- [Browser-Based Caricamenti tramite POST \(AWS Signature \(versione 4\)\)](#)

 Important

Leggi quanto segue sull'autenticazione e il controllo degli accessi prima di passare ad argomenti specifici sulle API.

Le richieste ad Amazon S3 possono essere autenticate o anonime. L'accesso autenticato richiede credenziali che AWS possono essere utilizzate per autenticare le tue richieste.

Consigli per le chiamate API

Effettuare chiamate API REST direttamente dal codice può essere complicato. È necessario scrivere il codice necessario per calcolare una firma valida per autenticare le richieste. Consigliamo invece le seguenti alternative:

- Usa gli [AWS SDK](#) per inviare le tue richieste.

Inoltre, consulta il codice [di esempio e le librerie](#).

Se utilizzi gli SDK, non è necessario scrivere codice per calcolare una firma per l'autenticazione delle richieste perché i client SDK autenticano le tue richieste utilizzando le chiavi di accesso fornite. A meno che tu non abbia una buona ragione per non farlo, dovresti sempre usare gli SDK. AWS

- Usa l'interfaccia a AWS riga di comando per effettuare chiamate API Amazon S3. Per informazioni sulla configurazione della AWS CLI ed esempi di comandi Amazon S3, consulta i seguenti argomenti:

[Configura l'interfaccia a riga di AWS comando](#) nella Guida per l'utente di Amazon Simple Storage Service.

[Utilizzo di Amazon S3 con l'interfaccia a riga di AWS comando](#) nella Guida per l'AWS Command Line Interface utente.

Effettuare chiamate dirette all'API REST

Note

L'intestazione della richiesta PUT è limitata a una dimensione di 8 KB. Nell'intestazione della richiesta PUT, la dimensione dei metadati definiti dal sistema è limitata a 2 KB. La dimensione dei metadati definiti dal sistema viene misurata sommando il numero di byte nella US-ASCII codifica di ogni chiave e valore.

Se desideri effettuare chiamate API REST personalizzate invece di utilizzare una delle alternative precedenti, ci sono alcune cose da tenere a mente.

- Per effettuare chiamate dirette all'API REST dal tuo codice, crea una firma utilizzando credenziali valide e includi la firma nella richiesta. Per informazioni sui vari metodi di autenticazione e sul calcolo delle firme, consulta [Autenticazione delle richieste \(AWS Signature \(versione 4\)\)](#).
- L'API REST utilizza intestazioni HTTP e codici di stato standard, pertanto i browser e i toolkit standard funzionano come previsto. In alcune aree, abbiamo aggiunto funzionalità all'HTTP (ad esempio, abbiamo aggiunto intestazioni per supportare il controllo degli accessi). In questi casi, abbiamo aggiunto la nuova funzionalità in modo che corrisponda allo stile di utilizzo standard di HTTP. Per ulteriori informazioni sull'esecuzione di richieste, consulta [Esecuzione di richieste](#).

Permissions

Puoi disporre di credenziali valide per autenticare le tue richieste, ma a meno che tu non disponga delle autorizzazioni S3 del proprietario dell'account o del proprietario del bucket non puoi creare o accedere alle risorse Amazon S3. Queste autorizzazioni vengono generalmente concesse tramite una policy AWS Identity and Access Management (IAM), ad esempio una bucket [policy](#). Ad esempio, è necessario disporre delle autorizzazioni per creare un bucket S3 o ottenere un oggetto in un bucket. Per un elenco completo delle autorizzazioni S3, consulta [Azioni, risorse e chiavi di condizione per Amazon S3](#).

Per ulteriori informazioni sulle autorizzazioni alle operazioni API S3 per tipi di risorse S3, consulta Autorizzazioni [richieste per le operazioni API di Amazon S3](#) nella Amazon Simple Storage Service User Guide.

Se utilizzi le credenziali utente root del tuo account Account AWS, disponi di tutte le autorizzazioni. Tuttavia, non è consigliabile utilizzare le credenziali dell'utente root. Ti consigliamo invece di creare ruoli AWS Identity and Access Management (IAM) nel tuo account e gestire le autorizzazioni utente. Per ulteriori informazioni, consulta [Access Management](#) nella Amazon Simple Storage Service User Guide.

Sviluppo con Amazon S3

In questa sezione sono riportati gli argomenti relativi agli sviluppatori per l'uso di Amazon S3. Per ulteriori informazioni, consulta gli argomenti riportati di seguito.

Argomenti

- [Esecuzione di richieste](#)
- [Sviluppo con Amazon S3 utilizzando AWS CLI](#)
- [Sviluppo con Amazon S3 utilizzando AWS SDK](#)
- [Ottenere gli ID di richiesta Amazon S3 per Supporto AWS](#)
- [Operazioni API a livello di oggetto di Amazon S3 supportate per tabelle S3](#)

Esecuzione di richieste

Amazon S3 è un servizio REST. Puoi inviare richieste ad Amazon S3 utilizzando l'API REST o l' AWS SDK (vedi [Sample Code and Libraries](#)), librerie wrapper che racchiudono l'API REST di Amazon S3 sottostante, semplificando le attività di programmazione.

Ogni interazione con Amazon S3 è autenticata o anonima. L'autenticazione è un processo teso a verificare l'identità del richiedente che cerca di accedere a un prodotto Amazon Web Services (AWS). Le richieste autenticate devono includere un valore di firma che autentichi il mittente della richiesta. Il valore della firma è, in parte, generato dalle chiavi di accesso del richiedente (ID della chiave di AWS accesso e chiave di accesso segreta). Per ulteriori informazioni su come ottenere le chiavi di accesso, consulta [Come ottenere le credenziali di sicurezza?](#) in Riferimenti generali di AWS.

Se si utilizza l' AWS SDK, le librerie calcolano la firma in base alle chiavi fornite. Tuttavia, se vengono effettuate chiamate API REST direttamente dall'applicazione in uso, è necessario scrivere il codice per calcolare la firma e aggiungerlo alla richiesta.

Argomenti

- [Le chiavi di accesso](#)
- [Endpoint della richiesta](#)
- [Esecuzione di richieste ad Amazon S3 su IPv6](#)
- [Effettuare richieste utilizzando AWS SDK](#)

- [Esecuzione di richieste con l'utilizzo di API REST](#)

Le chiavi di accesso

Le sezioni seguenti esaminano i tipi di chiavi di accesso che è possibile utilizzare per effettuare richieste autenticate.

Account AWS chiavi di accesso

Le chiavi di accesso all'account forniscono l'accesso completo alle AWS risorse di proprietà dell'account. Di seguito vengono illustrati alcuni esempi di chiavi di accesso:

- ID chiave di accesso (una stringa alfanumerica di 20 caratteri). Ad esempio, AKIAIOSFODNN7EXAMPLE
- Secret Access Key (una stringa di 40 caratteri). Ad esempio, wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

L'ID della chiave di accesso identifica in modo univoco un Account AWS. È possibile utilizzare queste chiavi di accesso per inviare le richieste autenticate a Amazon S3.

Chiavi di accesso utente IAM

Puoi crearne uno Account AWS per la tua azienda; tuttavia, potrebbero esserci diversi dipendenti dell'organizzazione che hanno bisogno di accedere alle risorse dell'AWS organizzazione. La condivisione delle chiavi di Account AWS accesso riduce la sicurezza e la creazione di chiavi individuali Account AWS per ogni dipendente potrebbe non essere pratica. Inoltre, non è facile condividere risorse quali i bucket e gli oggetti in quanto sono di proprietà di diversi account. Per condividere le risorse, è necessario assegnare le autorizzazioni, il che comporta un lavoro aggiuntivo.

In questi scenari, puoi utilizzare AWS Identity and Access Management (IAM) per creare utenti Account AWS con le proprie chiavi di accesso e allegare politiche utente IAM che concedono a tali utenti le autorizzazioni di accesso alle risorse appropriate. Per gestire meglio tali utenti, IAM consente la creazione di gruppi di utenti e l'assegnazione di autorizzazioni a livello di gruppo valide per tutti gli utenti presenti in quel gruppo.

Questi utenti sono denominati utenti IAM creati e gestiti all'interno dell'AWS utente. L'account padre verifica la capacità di un utente di accedere ad AWS. Qualsiasi risorsa creata da un utente IAM ricade sotto il controllo dell'Account AWS padre e viene pagata da quest'ultimo. Questi utenti IAM possono

inviare richieste autenticate a Amazon S3 utilizzando le loro credenziali di sicurezza. Per ulteriori informazioni sulla creazione e la gestione degli utenti all'interno del tuo Account AWS, vai alla pagina dei dettagli [AWS Identity and Access Management del prodotto](#).

Credenziali di sicurezza temporanee

Oltre a creare utenti IAM con le proprie chiavi di accesso, IAM consente anche di concedere credenziali di sicurezza temporanee (chiavi di accesso temporanee e un token di sicurezza) a qualsiasi utente IAM in modo che possa accedere ai tuoi AWS servizi e alle tue risorse. È inoltre possibile gestire gli utenti nel sistema in uso al di fuori di AWS. Tali risorse sono denominate utenti federati. Inoltre, gli utenti possono essere applicazioni create per accedere alle AWS risorse.

IAM fornisce l' Servizio di token di sicurezza AWS API per richiedere credenziali di sicurezza temporanee. Puoi utilizzare l'API AWS STS o l' AWS SDK per richiedere queste credenziali. L'API restituisce le credenziali di sicurezza (ID chiave di accesso e Secret Access Key) e un token di sicurezza. Queste credenziali sono valide solo per la durata specificata al momento della richiesta. L'ID chiave di accesso e la chiave segreta si utilizzano nello stesso modo in cui vengono impiegati quando si inviano le richieste usando l' Account AWS o le chiavi di accesso dell'utente IAM. Inoltre, è necessario includere il token in ogni richiesta inviata a Amazon S3.

Un utente IAM può richiedere queste credenziali di sicurezza temporanee per uso personale oppure per passarle agli utenti federati o alle applicazioni. Quando si effettua la richiesta di credenziali di sicurezza per gli utenti federati, è necessario fornire un nome utente e una policy IAM che definisce le autorizzazioni che si vuole associare a tali credenziali di sicurezza. L'utente federato non può avere un numero di autorizzazioni superiore a quello di cui dispone l'utente IAM padre che ha richiesto le credenziali di sicurezza.

È possibile utilizzare queste credenziali di sicurezza temporanee quando si effettuano richieste ad Amazon S3. Le librerie API calcolano il valore di firma necessario utilizzando tali credenziali per autenticare la richiesta dell'utente. In caso di invio di richieste tramite l'utilizzo di credenziali scadute, Amazon S3 rifiuta la richiesta.

Per informazioni sulla firma delle richieste mediante l'utilizzo di credenziali di sicurezza temporanee nelle richieste API REST, consulta [Firma e autenticazione delle richieste REST \(AWS firma \(versione 2\)\)](#). Per informazioni sull'invio di richieste tramite AWS SDK, consulta. [Effettuare richieste utilizzando AWS SDK](#)

Per ulteriori informazioni sul supporto IAM per le credenziali di sicurezza temporanee, consulta la sezione relativa alle [credenziali di sicurezza temporanee](#) nella Guida per l'utente di IAM.

Per maggiore sicurezza, è possibile richiedere l'autenticazione a più fattori (MFA) quando si effettua l'accesso alle risorse Amazon S3 configurando una policy del bucket. Per informazioni, consulta [Esempio di politiche dei bucket: Richiedere l'MFA](#). Una volta richiesta l'MFA per accedere alle risorse Amazon S3, il solo modo per accedervi è fornendo le credenziali temporanee che vengono create con una chiave MFA. Per ulteriori informazioni, consulta la pagina dei [dettagli dell'AWS Multi-Factor authentication](#) e [Configurazione dell'accesso alle MFA-Protected API](#) nella IAM User Guide.

Endpoint della richiesta

Le richieste REST vengono inviate all'endpoint predefinito del servizio. Per un elenco di tutti i AWS servizi e dei relativi endpoint, vai a [Regioni ed endpoint](#) in. Riferimenti generali di AWS

Esecuzione di richieste ad Amazon S3 su IPv6

Amazon Simple Storage Service (Amazon S3) consente di accedere ai bucket S3 tramite l'Internet Protocol versione 6 (IPv6), oltre al protocollo IPv4. Gli endpoint dual-stack Amazon S3; supportano le richieste ai bucket S3 su IPv6 e IPv4. Non sono previsti costi aggiuntivi per l'accesso ad Amazon S3 su IPv6. Per ulteriori informazioni sui prezzi, consulta [Prezzi di Amazon S3](#).

Note

Le seguenti informazioni si riferiscono agli endpoint IPv6 pubblici di Amazon S3. Per informazioni sugli endpoint AWS Virtual Private Cloud (VPC), consulta Amazon S3. [AWS PrivateLink](#)

Argomenti

- [Nozioni di base sull'esecuzione di richieste su IPv6](#)
- [Utilizzo degli indirizzi IPv6 nelle policy IAM](#)
- [Test di compatibilità degli indirizzi IP](#)
- [Utilizzo degli endpoint dual-stack Amazon S3](#)

Nozioni di base sull'esecuzione di richieste su IPv6

Per effettuare una richiesta a un bucket S3 su IPv6, è necessario utilizzare un endpoint dual-stack. Nella sezione seguente viene descritto come effettuare richieste su IPv6 utilizzando gli endpoint dual-stack.

Di seguito sono descritti alcuni punti da prendere in considerazione prima di tentare di accedere a un bucket su IPv6:

- Il client e la rete che eseguono l'accesso al bucket devono essere abilitati a utilizzare IPv6.
- Per l'accesso a IPv6 sono supportate sia le richieste in stile percorso sia quelle in stile hosting virtuale. Per ulteriori informazioni, consulta [Endpoint dual-stack Amazon S3](#).
- Se utilizzi il filtro degli indirizzi IP di origine nelle tue policy utente o bucket AWS Identity and Access Management (IAM), devi aggiornare le policy per includere gli intervalli di indirizzi IPv6. Per ulteriori informazioni, consulta [Utilizzo degli indirizzi IPv6 nelle policy IAM](#).
- Quando si utilizza IPv6, i file di log degli accessi al server generano indirizzi IP in un formato IPv6. È necessario aggiornare il software, gli script e gli strumenti esistenti utilizzati per analizzare i file di log di Amazon S3; affinché possano analizzare gli indirizzi Remote IP in formato IPv6. Per ulteriori informazioni, consulta [Registrazione delle richieste con registrazione degli accessi al server](#).

 Note

Se si verificano problemi relativi alla presenza di indirizzi IPv6 nei file di log, contatta [Supporto AWS](#).

Esecuzione di richieste su IPv6 tramite gli endpoint dual-stack

È possibile effettuare richieste con chiamate all'API di Amazon S3 tramite IPv6 utilizzando gli endpoint dual-stack. Le operazioni dell'API di Amazon S3 vengono eseguite nello stesso modo sia che tu acceda a Amazon S3 su IPv6 che su IPv4. Anche le prestazioni dovrebbero essere le stesse.

Quando si utilizza l'API REST, è possibile accedere direttamente a un endpoint dual-stack. Per ulteriori informazioni, consulta [Dual-stack punti finali](#).

Quando si utilizzano AWS Command Line Interface (AWS CLI) e AWS SDK, è possibile utilizzare un parametro o un flag per passare a un endpoint dual-stack. È inoltre possibile specificare l'endpoint dual-stack direttamente come sostituzione dell'endpoint Amazon S3 nel file di configurazione.

È possibile utilizzare un endpoint dual-stack per accedere a un bucket su IPv6 da uno qualsiasi dei seguenti elementi:

- Poi, vedi. AWS CLI [Utilizzo degli endpoint dual-stack di AWS CLI](#)

- Gli AWS SDK, vedete [Utilizzo degli endpoint dual-stack di AWS SDK](#).
- API REST, consulta [Esecuzione di richieste a endpoint Dual-Stack utilizzando l'API REST](#).

Caratteristiche non disponibili su IPv6

La seguente funzionalità non è attualmente supportata quando si accede a un bucket S3 su IPv6: sito Web statico hosting da un bucket S3.

Utilizzo degli indirizzi IPv6 nelle policy IAM

Prima di tentare di accedere a un bucket tramite IPv6-S3, è necessario assicurarsi che le policy utente IAM o del bucket IPv6 utilizzate per il filtro degli indirizzi IP siano aggiornate per includere gli intervalli di indirizzi IPv6. Se non aggiorni le policy di filtro degli indirizzi IP per la gestione degli indirizzi IPv6, i client possono perdere oppure ottenere in modo errato l'accesso al bucket quando iniziano a utilizzare IPv6. Per ulteriori informazioni sulla gestione delle autorizzazioni di accesso con IAM, consulta [Gestione delle identità e degli accessi per Amazon S3](#).

Le policy IAM che filtrano gli indirizzi IP utilizzano gli [operatori di condizione degli indirizzi IP](#). La seguente policy del bucket identifica l'intervallo 54.240.143.* degli indirizzi IPv4 consentiti tramite gli operatori di condizione degli indirizzi IP. A qualsiasi indirizzo IP non incluso in questo intervallo verrà negato l'accesso al bucket (examplebucket). Poiché tutti gli indirizzi IPv6 non sono inclusi nell'intervallo consentito, questa policy impedisce agli indirizzi IPv6 di accedere a examplebucket.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::examplebucket/*",
      "Condition": {
        "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
      }
    }
  ]
}
```

```
}
```

È possibile modificare l'elemento `Condition` della policy del bucket per consentire entrambi gli intervalli di indirizzi IPv4 (54.240.143.0/24) e IPv6 (2001:DB8:1234:5678::/64), come mostrato nell'esempio che segue. È possibile utilizzare lo stesso tipo di blocco `Condition` mostrato nell'esempio per aggiornare sia le policy del bucket sia le policy utente IAM.

```
"Condition": {
  "IpAddress": {
    "aws:SourceIp": [
      "54.240.143.0/24",
      "2001:DB8:1234:5678::/64"
    ]
  }
}
```

Prima di utilizzare IPv6 è necessario aggiornare tutte le policy relative agli utenti e ai bucket IAM che utilizzano il filtro degli indirizzi IP. Evita di utilizzare il filtro degli indirizzi IP nelle policy dei bucket.

Puoi rivedere le tue politiche utente IAM utilizzando la console IAM all'indirizzo. <https://console.aws.amazon.com/iam/> Per ulteriori informazioni su IAM, consulta la [Guida per l'utente di IAM](#). Per informazioni sulla modifica delle policy dei bucket S3, vedi [Aggiungere una policy sui bucket](#).

Test di compatibilità degli indirizzi IP

Se utilizzi Linux/Unix o Mac OS X, puoi verificare se puoi accedere a un endpoint dual-stack tramite IPv6 utilizzando il comando come mostrato nell'esempio seguente: `curl`

Example

```
curl -v http://s3.dualstack.us-west-2.amazonaws.com/
```

Vengono restituite informazioni simili a quelle indicate nell'esempio seguente. Se la connessione è stata eseguita su IPv6, l'indirizzo IP connesso sarà un indirizzo IPv6.

```
* About to connect() to s3-us-west-2.amazonaws.com port 80 (#0)
* Trying IPv6 address... connected
* Connected to s3.dualstack.us-west-2.amazonaws.com (IPv6 address) port 80 (#0)
```

```
> GET / HTTP/1.1
> User-Agent: curl/7.18.1 (x86_64-unknown-linux-gnu) libcurl/7.18.1 OpenSSL/1.0.1t
  zlib/1.2.3
> Host: s3.dualstack.us-west-2.amazonaws.com
```

Se utilizzi Microsoft 7 o 10, puoi verificare se puoi accedere a un endpoint dual-stack tramite IPv6 o IPv4 utilizzando il comando come mostrato nell'esempio seguente. `ping`

```
ping ipv6.s3.dualstack.us-west-2.amazonaws.com
```

Utilizzo degli endpoint dual-stack Amazon S3

Gli endpoint dual-stack Amazon S3; supportano le richieste ai bucket S3 su IPv6 e IPv4. In questa sezione viene descritto come utilizzare gli endpoint dual-stack.

Argomenti

- [Endpoint dual-stack Amazon S3](#)
- [Utilizzo degli endpoint dual-stack di AWS CLI](#)
- [Utilizzo degli endpoint dual-stack di AWS SDK](#)
- [Utilizzo degli endpoint dual-stack dall'API REST](#)

Endpoint dual-stack Amazon S3

Quando effettui una richiesta a un endpoint dual-stack, l'URL del bucket restituisce un indirizzo IPv6 o IPv4. Per ulteriori informazioni sull'accesso a un bucket su IPv6, consulta [Esecuzione di richieste ad Amazon S3 su IPv6](#).

Quando si utilizza l'API REST, è possibile accedere direttamente a un endpoint Amazon S3 utilizzando il nome dell'endpoint (URI). È possibile accedere a un bucket S3 tramite un endpoint dual-stack utilizzando un nome di endpoint in stile hosting virtuale o in stile percorso. Amazon S3 supporta solo i nomi di endpoint dual-stack regionali, il che significa che è necessario specificare la regione come parte del nome.

Utilizzare le convenzioni di denominazione seguenti per i nomi di endpoint dual-stack in stile hosting virtuale e in stile percorso.

- Endpoint dual-stack in stile hosting virtuale:

bucketname.s3.dualstack. *aws-region*.amazonaws.com

- Path-style endpoint dual-stack:

s3.dualstack. *aws-region*.amazonaws.com/ *bucketname*

Per ulteriori informazioni sullo stile dei nomi degli endpoint, consulta [Accedere ed elencare un bucket Amazon S3](#). Per un elenco di endpoint di Amazon S3, consultare la sezione relativa a [regioni ed endpoint](#) nella Riferimenti generali di AWS.

Important

È possibile utilizzare Transfer Acceleration con gli endpoint dual-stack. Per ulteriori informazioni, consulta [Guida introduttiva ad Amazon S3 Transfer Acceleration](#).

Note

I due tipi di endpoint Virtual Private Cloud (VPC) che accedono ad Amazon S3 (endpoint Interface VPC ed endpoint Gateway VPC) ora dispongono del supporto dual-stack. Per ulteriori informazioni sugli endpoint VPC per Amazon S3, consulta Amazon S3. [AWS PrivateLink](#)

Quando usi AWS Command Line Interface (AWS CLI) e AWS SDK, puoi utilizzare un parametro o un flag per passare a un endpoint dual-stack. È inoltre possibile specificare l'endpoint dual-stack direttamente come sostituzione dell'endpoint Amazon S3 nel file di configurazione. Le sezioni seguenti descrivono come utilizzare gli endpoint dual-stack degli SDK e degli SDK. AWS CLI AWS

Utilizzo degli endpoint dual-stack di AWS CLI

Questa sezione fornisce esempi di AWS CLI comandi utilizzati per effettuare richieste a un endpoint dual-stack. Per istruzioni sulla configurazione di, vedere. AWS CLI [Sviluppo con Amazon S3 utilizzando AWS CLI](#)

Imposta il valore `use_dualstack_endpoint` di configurazione su `true` in un profilo nel tuo AWS Config file per indirizzare tutte le richieste Amazon S3 effettuate dai `s3api` AWS CLI comandi `s3` and all'endpoint dual-stack per la regione specificata. La regione va specificata nel file di configurazione o in un comando tramite l'opzione `--region`.

Quando si utilizzano endpoint dual-stack con, sono supportati sia gli stili di indirizzamento che AWS CLI quelli. `path virtual` Lo stile di indirizzamento, impostato nel file di configurazione, controlla se il nome del bucket si trova nel nome host o fa parte dell'URL. Per default, l'interfaccia a riga di comando (CLI) tenta di utilizzare lo stile virtuale, se possibile, tornando allo stile percorso, se necessario. Per ulteriori informazioni, consulta [Configurazione di Amazon S3 per AWS CLI](#).

È anche possibile apportare modifiche alla configurazione tramite un comando, come mostrato nell'esempio seguente, dove `use_dualstack_endpoint` viene impostato su `true` e `addressing_style` viene impostato su `virtual` nel profilo di default.

```
$ aws configure set default.s3.use_dualstack_endpoint true
$ aws configure set default.s3.addressing_style virtual
```

Se desideri utilizzare un endpoint dual-stack solo per AWS CLI comandi specifici (non per tutti i comandi), puoi utilizzare uno dei seguenti metodi:

- È possibile utilizzare l'endpoint dual-stack per comando, impostando il parametro `--endpoint-url` su `https://s3.dualstack.aws-region.amazonaws.com` o `http://s3.dualstack.aws-region.amazonaws.com` per un comando `s3` o `s3api`.

```
$ aws s3api list-objects --bucket bucketname --endpoint-url https://s3.dualstack.aws-region.amazonaws.com
```

- Puoi impostare profili separati nel tuo file. AWS Config Ad esempio, si può creare un profilo che imposta `use_dualstack_endpoint` su `true` e un profilo che non imposta `use_dualstack_endpoint`. Quando si esegue un comando, specificare il profilo da usare, a seconda se si desidera o meno utilizzare l'endpoint dual-stack.

Note

Quando si utilizza il attualmente non AWS CLI è possibile utilizzare l'accelerazione di trasferimento con endpoint dual-stack. Tuttavia, il supporto per il sarà presto disponibile. AWS CLI Per ulteriori informazioni, consulta [Abilitazione e utilizzo di S3 Transfer Acceleration](#).

Utilizzo degli endpoint dual-stack di AWS SDK

Questa sezione fornisce esempi di come accedere a un endpoint dual-stack utilizzando gli SDK. AWS

AWS SDK per Java esempio di endpoint dual-stack

Nell'esempio seguente viene mostrato come abilitare endpoint dual-stack durante la creazione di un client Amazon S3 utilizzando la AWS SDK per Java.

Per istruzioni su come creare e testare un esempio Java funzionante, vedi [Guida introduttiva](#) nella Developer Guide. AWS SDK per Java

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;

public class DualStackEndpoints {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            // Create an Amazon S3 client with dual-stack endpoints enabled.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .withDualstackEnabled(true)
                .build();

            s3Client.listObjects(bucketName);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Se si utilizza Windows, potrebbe essere necessario impostare la seguente proprietà della macchina virtuale Java (JVM): AWS SDK per Java

```
java.net.preferIPv6Addresses=true
```

AWS Esempio di endpoint dual-stack .NET SDK

Quando si utilizza l' AWS SDK per .NET, si utilizza la `AmazonS3Config` classe per abilitare l'uso di un endpoint dual-stack, come mostrato nell'esempio seguente.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class DualStackEndpointTest
    {
        private const string bucketName = "**** bucket name ****";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            var config = new AmazonS3Config
            {
                UseDualstackEndpoint = true,
                RegionEndpoint = bucketRegion
            };
            client = new AmazonS3Client(config);
            Console.WriteLine("Listing objects stored in a bucket");
            ListingObjectsAsync().Wait();
        }

        private static async Task ListingObjectsAsync()
        {
            try
            {
                var request = new ListObjectsV2Request
                {
```

```
        BucketName = bucketName,
        MaxKeys = 10
    };
    ListObjectsV2Response response;
    do
    {
        response = await client.ListObjectsV2Async(request);

        // Process the response.
        foreach (S3Object entry in response.S3Objects)
        {
            Console.WriteLine("key = {0} size = {1}",
                              entry.Key, entry.Size);
        }
        Console.WriteLine("Next Continuation Token: {0}",
                           response.NextContinuationToken);
        request.ContinuationToken = response.NextContinuationToken;
    } while (response.IsTruncated == true);
    }
    catch (AmazonS3Exception amazonS3Exception)
    {
        Console.WriteLine("An AmazonS3Exception was thrown. Exception: " +
                           amazonS3Exception.ToString());
    }
    catch (Exception e)
    {
        Console.WriteLine("Exception: " + e.ToString());
    }
}
}
```

Per informazioni sulla configurazione e l'esecuzione degli esempi di codice, consulta [Getting Started with the AWS SDK for .NET](#) nella SDK for .NET AWS Developer Guide.

Utilizzo degli endpoint dual-stack dall'API REST

Per informazioni su come eseguire richieste su endpoint dual-stack tramite l'API REST, consulta [Esecuzione di richieste a endpoint Dual-Stack utilizzando l'API REST](#).

Effettuare richieste utilizzando AWS SDK

Argomenti

- [Effettuare richieste utilizzando Account AWS o credenziali utente IAM](#)
- [Esecuzione di richieste mediante le credenziali temporanee per gli utenti IAM](#)
- [Esecuzione di richieste mediante le credenziali temporanee per gli utenti federati](#)

Puoi inviare richieste autenticate ad Amazon S3 utilizzando l' AWS SDK o effettuando le chiamate API REST direttamente nella tua applicazione. L'API AWS SDK utilizza le credenziali fornite per calcolare la firma per l'autenticazione. Se si utilizza l'API REST direttamente nelle applicazioni, è necessario scrivere il codice necessario per calcolare la firma per l'autenticazione della richiesta. Per un elenco degli AWS SDK disponibili, vai a [Sample Code and Libraries](#).

Effettuare richieste utilizzando Account AWS o credenziali utente IAM

Puoi utilizzare le tue credenziali di sicurezza Account AWS o quelle degli utenti IAM per inviare richieste autenticate ad Amazon S3. Questa sezione fornisce esempi di come inviare richieste autenticate utilizzando i AWS SDK per Java, e. AWS SDK per .NET, AWS SDK per PHP. Per un elenco degli AWS SDK disponibili, vai a [Sample Code and Libraries](#).

Ciascuno di questi AWS SDK utilizza una catena di fornitori di SDK-specific credenziali per trovare e utilizzare le credenziali ed eseguire azioni per conto del proprietario delle credenziali. Ciò che accomuna tutte queste catene di fornitori di credenziali è che cercano tutte il tuo file di credenziali locale. AWS

Per ulteriori informazioni, consulta gli argomenti riportati di seguito.

Argomenti

- [Per creare un locale AWS file delle credenziali](#)
- [Invio di richieste autenticate utilizzando AWS SDK](#)
- [Risorse correlate](#)

Per creare un locale AWS file delle credenziali

Il modo più semplice per configurare le credenziali per i tuoi AWS SDK è utilizzare un AWS file di credenziali. Se usi AWS Command Line Interface (AWS CLI), potresti avere già configurato un file

di AWS credenziali locale. Altrimenti, per impostare un file delle credenziali, attenersi alla procedura riportata di seguito.

1. Accedi Console di gestione AWS e apri la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Creare un nuovo utente con autorizzazioni limitate ai servizi e alle operazioni a cui si desidera che il codice abbia accesso. Per ulteriori informazioni sulla creazione di un nuovo utente, consulta [Creazione di utenti IAM \(Console\)](#) e segui le istruzioni descritte fino al passaggio 8.
3. Scegliere Scarica .csv per salvare una copia locale delle credenziali AWS .
4. Sul computer, accedere alla directory principale e creare una directory .aws. Su Unix-based sistemi come Linux o OS X, si trova nella seguente posizione:

```
~/ .aws
```

In Windows, si trova nella seguente posizione:

```
%HOMEPATH%\ .aws
```

5. Nella directory .aws creare un nuovo file denominato `credentials`.
6. Aprire il file delle credenziali .csv scaricato dalla console IAM e copiarne il contenuto nel file `credentials` utilizzando il seguente formato:

```
[default]
aws_access_key_id = your_access_key_id
aws_secret_access_key = your_secret_access_key
```

7. Salvare il file `credentials` ed eliminare il file .csv scaricato al passaggio 3.

Il file delle credenziali condivise è ora configurato sul computer locale ed è pronto per essere utilizzato con gli AWS SDK.

Invio di richieste autenticate utilizzando AWS SDK

Usa gli AWS SDK per inviare richieste autenticate. Per ulteriori informazioni sull'invio di richieste autenticate, consulta [Credenziali di sicurezza AWS](#) o [Autenticazione IAM Identity Center](#).

Java

Per informazioni sull'autenticazione delle richieste tramite l' AWS SDK per Java, vedi [Utilizzo di file di configurazione e credenziali condivisi per configurare globalmente SDK e strumenti e Autenticazione e accesso tramite AWS SDK e strumenti nella Guida di riferimento per AWS SDK e strumenti](#). AWS

.NET

Per inviare richieste autenticate utilizzando le tue credenziali utente o IAM: Account AWS

- Crea un'istanza della classe `AmazonS3Client`.
- Eseguire uno dei metodi `AmazonS3Client` per inviare le richieste ad Amazon S3. Il client genera la firma necessaria dalle credenziali fornite e la include nella richiesta inviata ad Amazon S3.

Per ulteriori informazioni, consulta [Effettuare richieste utilizzando le credenziali AWS](#) dell'account o dell'utente IAM >.

Note

- È possibile creare il client `AmazonS3Client` senza specificare le credenziali di sicurezza. Le richieste inviate mediante questo client sono anonime e senza firma. Amazon S3 restituisce un errore se si inviano richieste anonime per una risorsa non disponibile pubblicamente.
- Puoi creare Account AWS e creare gli utenti richiesti. È inoltre possibile gestire le credenziali per tali utenti. Queste credenziali sono necessarie per eseguire l'attività nell'esempio seguente. Per ulteriori informazioni, consulta [Configurazione delle credenziali di AWS](#) nella Guida per gli sviluppatori di SDK per .NET .

È quindi possibile configurare l'applicazione per recuperare attivamente profili e credenziali e quindi utilizzare esplicitamente tali credenziali durante la creazione di un client di servizio. AWS Per ulteriori informazioni, consulta [Accesso a credenziali e profili in un'applicazione](#) nella Guida per gli sviluppatori di SDK per .NET .

L'esempio seguente di codice C# mostra come eseguire le precedenti operazioni. Per informazioni sulla configurazione e l'esecuzione degli esempi di codice, consulta [Getting Started with the AWS SDK for .NET](#) nella SDK for .NET AWS Developer Guide.

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class MakeS3RequestTest
    {
        private const string bucketName = "**** bucket name ****";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
        RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            using (client = new AmazonS3Client(bucketRegion))
            {
                Console.WriteLine("Listing objects stored in a bucket");
                ListingObjectsAsync().Wait();
            }
        }

        static async Task ListingObjectsAsync()
        {
            try
            {
                ListObjectsRequest request = new ListObjectsRequest
                {
                    BucketName = bucketName,
                    MaxKeys = 2
                };
                do
                {

```

```

        ListObjectsResponse response = await
client.ListObjectsAsync(request);
        // Process the response.
        foreach (S3Object entry in response.S3Objects)
        {
            Console.WriteLine("key = {0} size = {1}",
                entry.Key, entry.Size);
        }

        // If the response is truncated, set the marker to get the next
        // set of keys.
        if (response.IsTruncated)
        {
            request.Marker = response.NextMarker;
        }
        else
        {
            request = null;
        }
    } while (request != null);
}
catch (AmazonS3Exception e)
{
    Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
}
catch (Exception e)
{
    Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
}
}
}
}

```

PHP

Questa sezione spiega come utilizzare una classe della versione 3 per inviare richieste autenticate utilizzando AWS SDK per PHP le tue credenziali Account AWS o quelle dell'utente IAM. Per ulteriori informazioni sull'API AWS SDK for Ruby, vai a [AWS SDK for Ruby - Version 2.](#)

Il seguente esempio di codice PHP mostra come il client esegue una richiesta utilizzando le credenziali di sicurezza per elencare tutti i bucket per l'account.

Example

```
require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
]);

// Retrieve the list of buckets.
$result = $s3->listBuckets();

try {
    // Retrieve a paginator for listing objects.
    $objects = $s3->getPaginator('ListObjects', [
        'Bucket' => $bucket
    ]);

    echo "Keys retrieved!" . PHP_EOL;

    // Print the list of objects to the page.
    foreach ($objects as $object) {
        echo $object['Key'] . PHP_EOL;
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Note

È possibile creare il client `S3Client` senza specificare le credenziali di sicurezza. Le richieste inviate mediante questo client sono anonime e senza firma. Amazon S3 restituisce un errore se si inviano richieste anonime per una risorsa non disponibile

pubblicamente. Per ulteriori informazioni, vedere [Creazione di client anonimi](#) nella [Documentazione di AWS SDK per PHP](#).

Ruby

Prima di poter utilizzare la versione 3 dell'SDK AWS SDK per Ruby per effettuare chiamate ad Amazon S3, devi impostare le credenziali di AWS accesso utilizzate dall'SDK per verificare l'accesso ai tuoi bucket e oggetti. Se hai delle credenziali condivise impostate nel profilo delle AWS credenziali sul tuo sistema locale, la versione 3 dell'SDK per Ruby può utilizzare tali credenziali senza che tu debba dichiararle nel codice. Per ulteriori informazioni sulla configurazione delle credenziali condivise, consulta [Effettuare richieste utilizzando le credenziali dell'account o dell'utente IAM. AWS](#)

Il seguente frammento di codice Ruby utilizza le credenziali in un file di AWS credenziali condiviso su un computer locale per autenticare una richiesta e ottenere tutti i nomi delle chiavi degli oggetti in un bucket specifico. Esegue queste operazioni:

1. Crea un'istanza della classe `Aws::S3::Client`.
2. Effettua una richiesta ad Amazon S3 enumerando gli oggetti in un bucket utilizzando il metodo `list_objects_v2` di `Aws::S3::Client`. Il client genera il valore di firma necessario dalle credenziali contenute nel file delle AWS credenziali sul computer e lo include nella richiesta che invia ad Amazon S3.
3. Stampa la serie di nomi chiave degli oggetti al terminale.

Example

```
# Prerequisites:
# - An existing Amazon S3 bucket.

require 'aws-sdk-s3'

# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if all operations succeed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_bucket_objects?(s3_client, 'amzn-s3-demo-bucket')
```

```
def list_bucket_objects?(s3_client, bucket_name)
  puts "Accessing the bucket named '#{bucket_name}'..."
  objects = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if objects.count.positive?
    puts 'The object keys in this bucket are (first 50 objects):'
    objects.contents.each do |object|
      puts object.key
    end
  else
    puts 'No objects found in this bucket.'
  end

  true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
  false
end

# Example usage:
def run_me
  region = 'us-west-2'
  bucket_name = 'BUCKET_NAME'
  s3_client = Aws::S3::Client.new(region: region)

  exit 1 unless list_bucket_objects?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__
```

Se non disponi di un file di AWS credenziali locale, puoi comunque creare la `Aws::S3::Client` risorsa ed eseguire il codice su bucket e oggetti Amazon S3. Le richieste che vengono inviate utilizzando la versione 3 di SDK per Ruby sono richieste anonime, senza firma, per impostazione predefinita. Amazon S3 restituisce un errore se si inviano richieste anonime per una risorsa non disponibile pubblicamente.

Puoi utilizzare ed espandere il precedente snippet di codice per applicazioni SDK per Ruby, come nel seguente esempio, più complesso.

```
# Prerequisites:
# - An existing Amazon S3 bucket.

require 'aws-sdk-s3'

# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if all operations succeed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_bucket_objects?(s3_client, 'amzn-s3-demo-bucket')
def list_bucket_objects?(s3_client, bucket_name)
  puts "Accessing the bucket named '#{bucket_name}'..."
  objects = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if objects.count.positive?
    puts 'The object keys in this bucket are (first 50 objects):'
    objects.contents.each do |object|
      puts object.key
    end
  else
    puts 'No objects found in this bucket.'
  end

  true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
  false
end

# Example usage:
def run_me
  region = 'us-west-2'
  bucket_name = 'BUCKET_NAME'
  s3_client = Aws::S3::Client.new(region: region)

  exit 1 unless list_bucket_objects?(s3_client, bucket_name)
end
```

```
run_me if $PROGRAM_NAME == __FILE__
```

Go

Example

L'esempio seguente utilizza le AWS credenziali caricate automaticamente dall'SDK for Go dal file delle credenziali condivise.

```
package main

import (
    "context"
    "errors"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/smithy-go"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    s3Client := s3.NewFromConfig(sdkConfig)
    count := 10
    fmt.Printf("Let's list up to %v buckets for your account.\n", count)
    result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
    if err != nil {
        var ae smithy.APIError
        if errors.As(err, &ae) && ae.ErrorCode() == "AccessDenied" {
            fmt.Println("You don't have permission to list buckets for this account.")
        }
    }
}
```

```
    } else {
        fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
    }
    return
}
if len(result.Buckets) == 0 {
    fmt.Println("You don't have any buckets!")
} else {
    if count > len(result.Buckets) {
        count = len(result.Buckets)
    }
    for _, bucket := range result.Buckets[:count] {
        fmt.Printf("\t\t%v\n", *bucket.Name)
    }
}
}
```

Risorse correlate

- [Sviluppo con Amazon S3 utilizzando AWS SDK](#)
- [AWS SDK per PHP per Amazon S3 `Aws\S3\S3Client` Class](#)
- [Documentazione di AWS SDK per PHP](#)

Esecuzione di richieste mediante le credenziali temporanee per gli utenti IAM

Un utente IAM Account AWS o un utente IAM può richiedere credenziali di sicurezza temporanee e utilizzarle per inviare richieste autenticate ad Amazon S3. In questa sezione sono riportati esempi sull'utilizzo degli SDK AWS SDK per Java, .NET e PHP per ottenere credenziali di sicurezza temporanee con cui autenticare le richieste ad Amazon S3.

Java

Un utente o un IAM Account AWS può richiedere credenziali di sicurezza temporanee (vedi [Esecuzione di richieste](#)) utilizzando AWS SDK per Java e utilizzarle per accedere ad Amazon S3. Queste credenziali scadono al termine della durata della sessione specificata.

Per default, la sessione dura un'ora. Se si utilizzano le credenziali utente IAM, è possibile specificare la durata della richiesta delle credenziali di protezione temporanee da 15 minuti alla durata massima della sessione per il ruolo. Per ulteriori informazioni sulle credenziali di sicurezza temporanee, consulta la sezione relativa alle [credenziali di sicurezza temporanee](#) nella Guida per l'utente IAM. Per ulteriori informazioni sull'esecuzione di richieste, consulta [Esecuzione di richieste](#).

Per ottenere credenziali di sicurezza temporanee e accedere ad Amazon S3

1. Crea un'istanza della classe `AWSSecurityTokenService`.
2. Recuperare le credenziali di sicurezza temporanee per il ruolo desiderato chiamando il metodo `assumeRole()` del client Security Token Service (STS).
3. Creare un pacchetto di credenziali di sicurezza temporanee in un oggetto `BasicSessionCredentials`. Tale oggetto viene utilizzato per specificare le credenziali di sicurezza temporanee per il client Amazon S3.
4. Creare un'istanza della classe `AmazonS3Client` utilizzando le credenziali di sicurezza temporanee. Le richieste vengono inviate ad Amazon S3 con questo client. In caso di invio di richieste con credenziali scadute, Amazon S3 restituirà un errore.

Note

Se si ottengono credenziali di sicurezza temporanee utilizzando le credenziali di sicurezza dell' Account AWS , le credenziali temporanee rimarranno valide solo per un'ora. È

possibile specificare la durata della sessione solo se si utilizzano le credenziali utente Amazon S3 per richiedere una sessione.

L'esempio seguente elenca un set di chiavi degli oggetti incluse nel bucket specificato. L'esempio ottiene le credenziali di sicurezza temporanee per una sessione e le utilizza per inviare una richiesta autenticata ad Amazon S3.

Per testare l'esempio utilizzando le credenziali di un utente IAM, è necessario creare un utente IAM nell' Account AWS. Per ulteriori informazioni su come creare un utente IAM, consulta [Creazione del primo utente IAM e del gruppo di amministratori](#) nella Guida per l'utente di IAM.

Per istruzioni su come creare e testare un campione funzionante, consulta [Getting Started](#) nella AWS SDK per Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.AssumeRoleRequest;
import com.amazonaws.services.securitytoken.model.AssumeRoleResult;
import com.amazonaws.services.securitytoken.model.Credentials;

public class MakingRequestsWithIAMTempCredentials {
    public static void main(String[] args) {
        String clientRegion = "**** Client region ****";
        String roleARN = "**** ARN for role to be assumed ****";
        String roleSessionName = "**** Role session name ****";
        String bucketName = "**** Bucket name ****";

        try {
            // Creating the STS client is part of your trusted code. It has
            // the security credentials you use to obtain temporary security
            credentials.
```



```

        AWSSecurityTokenService stsClient =
AWSecurityTokenServiceClientBuilder.standard()
            .withCredentials(new ProfileCredentialsProvider())
            .withRegion(clientRegion)
            .build();

// Obtain credentials for the IAM role. Note that you cannot assume the
role of
// an AWS root account;
// Amazon S3 will deny access. You must use credentials for an IAM user
or an
// IAM role.
AssumeRoleRequest roleRequest = new AssumeRoleRequest()
    .withRoleArn(roleARN)
    .withRoleSessionName(roleSessionName);
AssumeRoleResult roleResponse = stsClient.assumeRole(roleRequest);
Credentials sessionCredentials = roleResponse.getCredentials();

// Create a BasicSessionCredentials object that contains the credentials
you
// just retrieved.
BasicSessionCredentials awsCredentials = new BasicSessionCredentials(
    sessionCredentials.getAccessKeyId(),
    sessionCredentials.getSecretAccessKey(),
    sessionCredentials.getSessionToken());

// Provide temporary security credentials so that the Amazon S3 client
// can send authenticated requests to Amazon S3. You create the client
// using the sessionCredentials object.
AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
    .withCredentials(new
AWSStaticCredentialsProvider(awsCredentials))
    .withRegion(clientRegion)
    .build();

// Verify that assuming the role worked and the permissions are set
correctly
// by getting a set of object keys from the bucket.
ObjectListing objects = s3Client.listObjects(bucketName);
System.out.println("No. of Objects: " +
objects.getObjectSummaries().size());
    } catch (AmazonServiceException e) {
        // The call was transmitted successfully, but Amazon S3 couldn't process
        // it, so it returned an error response.

```

```
        e.printStackTrace();
    } catch (SdkClientException e) {
        // Amazon S3 couldn't be contacted for a response, or the client
        // couldn't parse the response from Amazon S3.
        e.printStackTrace();
    }
}
```

.NET

Un utente o un IAM Account AWS può richiedere credenziali di sicurezza temporanee utilizzando AWS SDK per .NET e utilizzarle per accedere ad Amazon S3. Queste credenziali scadono al termine della durata della sessione.

Per default, la sessione dura un'ora. Se si utilizzano le credenziali utente IAM, è possibile specificare la durata della richiesta delle credenziali di protezione temporanee da 15 minuti alla durata massima della sessione per il ruolo. Per ulteriori informazioni sulle credenziali di sicurezza temporanee, consulta la sezione relativa alle [credenziali di sicurezza temporanee](#) nella Guida per l'utente IAM. Per ulteriori informazioni sull'esecuzione di richieste, consulta [Esecuzione di richieste](#).

Per ottenere credenziali di sicurezza temporanee e accedere ad Amazon S3

1. Crea un'istanza del Servizio di token di sicurezza AWS client, `AmazonSecurityTokenServiceClient`
2. Avviare una sessione richiamando il metodo `GetSessionToken` del client STS creato nella fase precedente. Le informazioni sulla sessione vengono fornite al metodo tramite un oggetto `GetSessionTokenRequest`.

Il metodo restituisce le credenziali di sicurezza temporanee.

3. Creare un pacchetto di credenziali di sicurezza temporanee in un'istanza dell'oggetto `SessionAWSCredentials`. Tale oggetto viene utilizzato per specificare le credenziali di sicurezza temporanee per il client Amazon S3.
4. Creare un'istanza della classe `AmazonS3Client` passando le credenziali di sicurezza temporanee. Le richieste vengono inviate ad Amazon S3 con questo client. In caso di invio di richieste mediante l'utilizzo di credenziali scadute, Amazon S3 restituisce un errore.

 Note

Se si ottengono credenziali di sicurezza temporanee utilizzando le credenziali di sicurezza dell' Account AWS , le credenziali temporanee rimarranno valide solo per un'ora.

Puoi specificare la durata di una sessione solo se utilizzi le credenziali utente IAM per richiedere una sessione.

L'esempio di codice C# seguente elenca le chiavi degli oggetti incluse nel bucket specificato. A titolo illustrativo, l'esempio ottiene le credenziali di sicurezza temporanee per una sessione di default della durata di un'ora e le utilizza per inviare una richiesta autenticata ad Amazon S3.

Per testare l'esempio utilizzando le credenziali di un utente IAM, è necessario creare un utente IAM nell' Account AWS. Per ulteriori informazioni su come creare un utente IAM, consulta [Creazione del primo utente IAM e del gruppo di amministratori](#) nella Guida per l'utente di IAM. Per ulteriori informazioni sull'esecuzione di richieste, consulta [Esecuzione di richieste](#).

Per informazioni sulla configurazione e l'esecuzione degli esempi di codice, consulta [Getting Started with the AWS SDK for .NET](#) nella AWS SDK for .NET Developer Guide.

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TempCredExplicitSessionStartTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
        RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {

```

```
        ListObjectsAsync().Wait();
    }

    private static async Task ListObjectsAsync()
    {
        try
        {
            // Credentials use the default AWS SDK for .NET credential search
chain.
            // On local development machines, this is your default profile.
            Console.WriteLine("Listing objects stored in a bucket");
            SessionAWSCredentials tempCredentials = await
GetTemporaryCredentialsAsync();

            // Create a client by providing temporary security credentials.
            using (s3Client = new AmazonS3Client(tempCredentials, bucketRegion))
            {
                var listObjectRequest = new ListObjectsRequest
                {
                    BucketName = bucketName
                };
                // Send request to Amazon S3.
                ListObjectsResponse response = await
s3Client.ListObjectsAsync(listObjectRequest);
                List<S3Object> objects = response.S3Objects;
                Console.WriteLine("Object count = {0}", objects.Count);
            }
        }
        catch (AmazonS3Exception s3Exception)
        {
            Console.WriteLine(s3Exception.Message, s3Exception.InnerException);
        }
        catch (AmazonSecurityTokenServiceException stsException)
        {
            Console.WriteLine(stsException.Message,
stsException.InnerException);
        }
    }

    private static async Task<SessionAWSCredentials>
GetTemporaryCredentialsAsync()
    {
        using (var stsClient = new AmazonSecurityTokenServiceClient())
        {
```

```
var getSessionTokenRequest = new GetSessionTokenRequest
{
    DurationSeconds = 7200 // seconds
};

GetSessionTokenResponse sessionTokenResponse =
    await
stsClient.GetSessionTokenAsync(getSessionTokenRequest);

Credentials credentials = sessionTokenResponse.Credentials;

var sessionCredentials =
    new SessionAWSCredentials(credentials.AccessKeyId,
                                credentials.SecretAccessKey,
                                credentials.SessionToken);

return sessionCredentials;
}
}
}
```

PHP

Per ulteriori informazioni sull'API AWS SDK for Ruby, vai a [AWS SDK for Ruby - Version 2](#).

Un utente o un utente IAM Account AWS può richiedere credenziali di sicurezza temporanee utilizzando la versione 3 di AWS SDK per PHP. Può quindi utilizzare tali credenziali per accedere ad Amazon S3. Le credenziali scadono alla fine della durata della sessione.

Per default, la sessione dura un'ora. Se si utilizzano le credenziali utente IAM, è possibile specificare la durata della richiesta delle credenziali di protezione temporanee da 15 minuti alla durata massima della sessione per il ruolo. Per ulteriori informazioni sulle credenziali di sicurezza temporanee, consulta la sezione relativa alle [credenziali di sicurezza temporanee](#) nella Guida per l'utente IAM. Per ulteriori informazioni sull'esecuzione di richieste, consulta [Esecuzione di richieste](#).

Note

Se si ottengono credenziali di sicurezza temporanee utilizzando le credenziali di sicurezza dell'Account AWS, le credenziali di sicurezza temporanee rimarranno valide solo per

un'ora. È possibile specificare la durata della sessione solo se si utilizzano le credenziali utente Amazon S3 per richiedere una sessione.

Example

L'esempio di codice PHP seguente elenca le chiavi degli oggetti incluse nel bucket specificato utilizzando credenziali di sicurezza temporanee. L'esempio ottiene le credenziali di sicurezza temporanee per una sessione di default della durata di un'ora e le utilizza per inviare una richiesta autenticata ad Amazon S3. Per ulteriori informazioni sull'API AWS SDK for Ruby, vai a [AWS SDK for Ruby - Version 2](#).

Per testare l'esempio utilizzando le credenziali di un utente IAM, è necessario creare un utente IAM nell' Account AWS. Per informazioni su come creare un utente IAM, consulta [Creazione del primo utente IAM e del gruppo di amministratori](#) nella Guida per l'utente IAM. Per degli esempi di impostazione della durata della sessione quando si utilizzano le credenziali utente IAM per richiedere una sessione, consulta [Esecuzione di richieste mediante le credenziali temporanee per gli utenti IAM](#).

```
require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$bucket = '*** Your Bucket Name ***';

$sts = new StsClient([
    'version' => 'latest',
    'region' => 'us-east-1'
]);

$sessionToken = $sts->getSessionToken();

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
    'credentials' => [
        'key' => $sessionToken['Credentials']['AccessKeyId'],
        'secret' => $sessionToken['Credentials']['SecretAccessKey'],
        'token' => $sessionToken['Credentials']['SessionToken']
    ]
]);
```

```
]
]);

$result = $s3->listBuckets();

try {
    // Retrieve a paginator for listing objects.
    $objects = $s3->getPaginator('ListObjects', [
        'Bucket' => $bucket
    ]);

    echo "Keys retrieved!" . PHP_EOL;

    // List objects
    foreach ($objects as $object) {
        echo $object['Key'] . PHP_EOL;
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Ruby

Un utente o un utente IAM Account AWS può richiedere credenziali di sicurezza temporanee utilizzandole AWS SDK per Ruby e utilizzarle per accedere ad Amazon S3. Queste credenziali scadono al termine della durata della sessione.

Per default, la sessione dura un'ora. Se si utilizzano le credenziali utente IAM, è possibile specificare la durata della richiesta delle credenziali di protezione temporanee da 15 minuti alla durata massima della sessione per il ruolo. Per ulteriori informazioni sulle credenziali di sicurezza temporanee, consulta la sezione relativa alle [credenziali di sicurezza temporanee](#) nella Guida per l'utente IAM. Per ulteriori informazioni sull'esecuzione di richieste, consulta [Esecuzione di richieste](#).

Note

Se si ottengono credenziali di sicurezza temporanee utilizzando le credenziali di sicurezza dell' Account AWS , le credenziali di sicurezza temporanee rimarranno valide solo per un'ora. Puoi specificare la durata della sessione solo se utilizzi le credenziali utente IAM per richiedere una sessione.

Nel seguente esempio di codice Ruby viene creato un utente temporaneo per elencare le voci in un bucket specificato per un'ora. Per utilizzare questo esempio, devi disporre di AWS credenziali con le autorizzazioni necessarie per creare nuovi Servizio di token di sicurezza AWS (AWS STS) client ed elencare i bucket Amazon S3.

```
# Prerequisites:
# - A user in AWS Identity and Access Management (IAM). This user must
#   be able to assume the following IAM role. You must run this code example
#   within the context of this user.
# - An existing role in IAM that allows all of the Amazon S3 actions for all of the
#   resources in this code example. This role must also trust the preceding IAM
#   user.
# - An existing S3 bucket.

require 'aws-sdk-core'
require 'aws-sdk-s3'
require 'aws-sdk-iam'

# Checks whether a user exists in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Boolean] true if the user exists; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   exit 1 unless user_exists?(iam_client, 'my-user')
def user_exists?(iam_client, user_name)
  response = iam_client.get_user(user_name: user_name)
  return true if response.user.user_name
rescue Aws::IAM::Errors::NoSuchEntity
  # User doesn't exist.
rescue StandardError => e
  puts 'Error while determining whether the user ' \
    "'#{user_name}' exists: #{e.message}"
end

# Creates a user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [AWS::IAM::Types::User] The new user.
# @example
```



```

#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   user = create_user(iam_client, 'my-user')
#   exit 1 unless user.user_name
def create_user(iam_client, user_name)
  response = iam_client.create_user(user_name: user_name)
  response.user
rescue StandardError => e
  puts "Error while creating the user '#{user_name}': #{e.message}"
end

# Gets a user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [AWS::IAM::Types::User] The existing user.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   user = get_user(iam_client, 'my-user')
#   exit 1 unless user.user_name
def get_user(iam_client, user_name)
  response = iam_client.get_user(user_name: user_name)
  response.user
rescue StandardError => e
  puts "Error while getting the user '#{user_name}': #{e.message}"
end

# Checks whether a role exists in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The role's name.
# @return [Boolean] true if the role exists; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   exit 1 unless role_exists?(iam_client, 'my-role')
def role_exists?(iam_client, role_name)
  response = iam_client.get_role(role_name: role_name)
  return true if response.role.role_name
rescue StandardError => e
  puts "Error while determining whether the role ' \
    '#{role_name}' exists: #{e.message}"
end

# Gets credentials for a role in IAM.
#

```

```

# @param sts_client [Aws::STS::Client] An initialized AWS STS client.
# @param role_arn [String] The role's Amazon Resource Name (ARN).
# @param role_session_name [String] A name for this role's session.
# @param duration_seconds [Integer] The number of seconds this session is valid.
# @return [AWS::AssumeRoleCredentials] The credentials.
# @example
#   sts_client = Aws::STS::Client.new(region: 'us-west-2')
#   credentials = get_credentials(
#     sts_client,
#     'arn:aws:iam::123456789012:role/AmazonS3ReadOnly',
#     'ReadAmazonS3Bucket',
#     3600
#   )
#   exit 1 if credentials.nil?
def get_credentials(sts_client, role_arn, role_session_name, duration_seconds)
  Aws::AssumeRoleCredentials.new(
    client: sts_client,
    role_arn: role_arn,
    role_session_name: role_session_name,
    duration_seconds: duration_seconds
  )
rescue StandardError => e
  puts "Error while getting credentials: #{e.message}"
end

# Checks whether a bucket exists in Amazon S3.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The name of the bucket.
# @return [Boolean] true if the bucket exists; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless bucket_exists?(s3_client, 'amzn-s3-demo-bucket')
def bucket_exists?(s3_client, bucket_name)
  response = s3_client.list_buckets
  response.buckets.each do |bucket|
    return true if bucket.name == bucket_name
  end
end
rescue StandardError => e
  puts "Error while checking whether the bucket '#{bucket_name}' " \
    "exists: #{e.message}"
end

# Lists the keys and ETags for the objects in an Amazon S3 bucket.

```

```
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if the objects were listed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_objects_in_bucket?(s3_client, 'amzn-s3-demo-bucket')
def list_objects_in_bucket?(s3_client, bucket_name)
  puts "Accessing the contents of the bucket named '#{bucket_name}'..."
  response = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if response.count.positive?
    puts "Contents of the bucket named '#{bucket_name}' (first 50 objects):"
    puts 'Name => ETag'
    response.contents.each do |obj|
      puts "#{obj.key} => #{obj.etag}"
    end
  else
    puts "No objects in the bucket named '#{bucket_name}'."
  end
  true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
end
```

Esecuzione di richieste mediante le credenziali temporanee per gli utenti federati

Puoi richiedere credenziali di sicurezza temporanee e fornirle agli utenti o alle applicazioni federati che devono accedere alle tue AWS risorse. Questa sezione fornisce esempi di come utilizzare l'AWS SDK per ottenere credenziali di sicurezza temporanee per utenti o applicazioni federati e inviare richieste autenticate ad Amazon S3 utilizzando tali credenziali. Per un elenco degli AWS SDK disponibili, consulta Codice e librerie di esempio. <https://aws.amazon.com/code/>

Note

Account AWS Sia l'utente IAM che un utente IAM possono richiedere credenziali di sicurezza temporanee per utenti federati. Tuttavia, per una maggiore sicurezza, la richiesta di credenziali temporanee è consentita solo a un utente IAM che dispone delle autorizzazioni necessarie, in modo da garantire che l'utente federato riceva esclusivamente le autorizzazioni dell'utente IAM che presenta la richiesta. In alcune applicazioni, può risultare utile creare un utente IAM con specifiche autorizzazioni finalizzate a concedere credenziali di accesso temporanee agli utenti federati e alle applicazioni.

Java

Puoi fornire credenziali di sicurezza temporanee per gli utenti e le applicazioni federati in modo che possano inviare richieste autenticate per accedere alle tue risorse. AWS Quando si richiedono le credenziali temporanee, è necessario fornire un nome utente e una policy IAM che descrive le autorizzazioni a livello di risorsa che si desidera concedere. Per default, la sessione dura un'ora. In fase di richiesta delle credenziali di sicurezza temporanee per utenti federati e applicazioni, è possibile impostare in modo esplicito un valore di durata diverso.

Note

Per una maggiore sicurezza nella richiesta di credenziali di sicurezza temporanee per utenti federati e applicazioni, consigliamo di utilizzare un utente IAM dedicato che abbia solo le autorizzazioni di accesso necessarie. L'utente temporaneo che si crea non può mai avere autorizzazioni più ampie rispetto all'utente IAM che ha richiesto le credenziali di sicurezza temporanee. Per ulteriori informazioni, consulta [FAQ AWS Identity and Access Management](#).

Per fornire credenziali di sicurezza e inviare una richiesta autenticata per accedere alle risorse, procedere come segue:

- Crea un'istanza della classe `AWSSecurityTokenServiceClient`.
- Avviare una sezione richiamando il metodo `getFederationToken()` del client Security Token Service (STS). Fornisci informazioni sulla sessione, incluso il nome utente e la policy IAM che si desidera collegare alle credenziali temporanee. È possibile specificare una durata della sessione opzionale. Questo metodo restituisce le credenziali di sicurezza temporanee.
- Creare un pacchetto di credenziali di sicurezza temporanee in un'istanza dell'oggetto `BasicSessionCredentials`. Tale oggetto viene utilizzato per specificare le credenziali di sicurezza temporanee per il client Amazon S3.
- Creare un'istanza della classe `AmazonS3Client` utilizzando le credenziali di sicurezza temporanee. Le richieste vengono inviate ad Amazon S3 con questo client. In caso di invio di richieste mediante l'utilizzo di credenziali scadute, Amazon S3 restituisce un errore.

Example

L'esempio elenca le chiavi nel bucket S3 specificato. Nell'esempio, è necessario ottenere le credenziali di sicurezza temporanee per l'utente federato per una sessione della durata di due ore e utilizzare le credenziali per inviare richieste autentiche ad Amazon S3. Per eseguire l'esempio, devi creare un utente IAM con una policy allegata che consenta all'utente di richiedere credenziali di sicurezza temporanee ed elencare le tue risorse. AWS Questa policy esegue quanto segue:

```
{
  "Statement": [{
    "Action": ["s3:ListBucket",
      "sts:GetFederationToken*"],
    "Effect": "Allow",
    "Resource": "*"
  }]
}
```

Per ulteriori informazioni su come creare un utente IAM, consulta [Creazione del primo utente IAM e del gruppo di amministratori](#) nella Guida per l'utente di IAM.

Dopo la creazione dell'utente IAM e il collegamento della policy di cui sopra, è possibile eseguire l'esempio seguente. Per istruzioni su come creare e testare un campione funzionante, consulta [Getting Started](#) nella AWS SDK per Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.Statement.Effect;
import com.amazonaws.auth.policy.actions.S3Actions;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.Credentials;
import com.amazonaws.services.securitytoken.model.GetFederationTokenRequest;
import com.amazonaws.services.securitytoken.model.GetFederationTokenResult;

import java.io.IOException;

public class MakingRequestsWithFederatedTempCredentials {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "**** Specify bucket name ****";
        String federatedUser = "**** Federated user name ****";
        String resourceARN = "arn:aws:s3:::" + bucketName;

        try {
            AWSSecurityTokenService stsClient = AWSSecurityTokenServiceClientBuilder
                .standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();
```

```
GetFederationTokenRequest getFederationTokenRequest = new
GetFederationTokenRequest();
getFederationTokenRequest.setDurationSeconds(7200);
getFederationTokenRequest.setName(federatedUser);

// Define the policy and add it to the request.
Policy policy = new Policy();
policy.withStatements(new Statement(Effect.Allow)
    .withActions(S3Actions.ListObjects)
    .withResources(new Resource(resourceARN)));
getFederationTokenRequest.setPolicy(policy.toJson());

// Get the temporary security credentials.
GetFederationTokenResult federationTokenResult =
stsClient.getFederationToken(getFederationTokenRequest);
Credentials sessionCredentials = federationTokenResult.getCredentials();

// Package the session credentials as a BasicSessionCredentials
// object for an Amazon S3 client object to use.
BasicSessionCredentials basicSessionCredentials = new
BasicSessionCredentials(
    sessionCredentials.getAccessKeyId(),
    sessionCredentials.getSecretAccessKey(),
    sessionCredentials.getSessionToken());
AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
    .withCredentials(new
AWSStaticCredentialsProvider(basicSessionCredentials))
    .withRegion(clientRegion)
    .build();

// To verify that the client works, send a listObjects request using
// the temporary security credentials.
ObjectListing objects = s3Client.listObjects(bucketName);
System.out.println("No. of Objects = " +
objects.getObjectSummaries().size());
} catch (AmazonServiceException e) {
    // The call was transmitted successfully, but Amazon S3 couldn't process
    // it, so it returned an error response.
    e.printStackTrace();
} catch (SdkClientException e) {
    // Amazon S3 couldn't be contacted for a response, or the client
    // couldn't parse the response from Amazon S3.
    e.printStackTrace();
}
```

```
}  
}
```

.NET

Puoi fornire credenziali di sicurezza temporanee per gli utenti e le applicazioni federati in modo che possano inviare richieste autenticate per accedere alle tue risorse. AWS Quando si richiedono le credenziali temporanee, è necessario fornire un nome utente e una policy IAM che descrive le autorizzazioni a livello di risorsa che si desidera concedere. Per impostazione predefinita, la sessione dura un'ora. In fase di richiesta delle credenziali di sicurezza temporanee per utenti federati e applicazioni, è possibile impostare in modo esplicito un valore di durata diverso. Per informazioni sull'invio di richieste autenticate, consulta [Esecuzione di richieste](#).

Note

Quando richiedi credenziali di sicurezza temporanee per utenti e applicazioni federati, per una maggiore sicurezza, utilizza un utente IAM dedicato con solo le autorizzazioni di accesso necessarie. L'utente temporaneo che si crea non può mai avere autorizzazioni più ampie rispetto all'utente IAM che ha richiesto le credenziali di sicurezza temporanee. Per ulteriori informazioni, consulta [FAQ AWS Identity and Access Management](#).

Esegui le operazioni indicate di seguito:

- Crea un'istanza del client, classe. Servizio di token di sicurezza AWS
`AmazonSecurityTokenServiceClient`
- Avviare una sessione chiamando il metodo `GetFederationToken` del client STS. Occorre fornire informazioni sulla sessione, incluso il nome utente e la policy IAM da collegare alle credenziali temporanee. Eventualmente, è possibile specificare una durata della sessione. Questo metodo restituisce le credenziali di sicurezza temporanee.
- Creare un pacchetto di credenziali di sicurezza temporanee in un'istanza dell'oggetto `SessionAWSCredentials`. Tale oggetto viene utilizzato per specificare le credenziali di sicurezza temporanee per il client Amazon S3.
- Creare un'istanza della classe `AmazonS3Client` passando le credenziali di sicurezza temporanee. Con questo client, le richieste vengono inviate ad Amazon S3. In caso di invio di richieste mediante l'utilizzo di credenziali scadute, Amazon S3 restituisce un errore.

Example

L'esempio C# seguente elenca le chiavi incluse nel bucket specificato. Nell'esempio, è necessario ottenere le credenziali di sicurezza temporanee per l'utente federato (User1) per una sessione della durata di due ore e utilizzare le credenziali per inviare richieste autenticate a Amazon S3.

- Per questo esercizio, si crea un utente IAM con le autorizzazioni minime. Utilizzando le credenziali di questo utente IAM, si richiedono le credenziali temporanee per gli altri. In questo esempio sono elencati solo gli oggetti di un bucket specifico. Crea un utente IAM con la seguente policy collegata:

```
{
  "Statement": [{
    "Action": ["s3:ListBucket",
      "sts:GetFederationToken*"],
    "Effect": "Allow",
    "Resource": "*"
  }]
}
```

La policy consente all'utente IAM di richiedere credenziali di sicurezza temporanee e autorizzazioni di accesso solo per elencare le AWS risorse. Per ulteriori informazioni su come creare un utente IAM, consulta [Creazione del primo utente IAM e del gruppo di amministratori](#) nella Guida per l'utente di IAM.

- Utilizza le credenziali di sicurezza dell'utente IAM per provare l'esempio seguente. Nell'esempio vengono inviate richieste autenticate ad Amazon S3 mediante le credenziali di sicurezza temporanee. Nell'esempio è specificata la seguente policy durante la richiesta di credenziali di sicurezza temporanee per l'utente federato (User1) con accesso limitato agli elenchi di oggetti di un bucket specifico (YourBucketName). Occorre specificare la policy e fornire il nome del bucket esistente.

```
{
  "Statement": [
    {
      "Sid": "1",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::YourBucketName"
    }
  ]
}
```

```
    }  
  ]  
}
```

- Example

Aggiorna il seguente esempio e fornire il nome del bucket specificato nella precedente policy di accesso dell'utente federato. Per informazioni sulla configurazione e l'esecuzione degli esempi di codice, consulta [Getting Started with the AWS SDK for .NET](#) nella SDK for .NET AWS Developer Guide.

```
using Amazon;  
using Amazon.Runtime;  
using Amazon.S3;  
using Amazon.S3.Model;  
using Amazon.SecurityToken;  
using Amazon.SecurityToken.Model;  
using System;  
using System.Collections.Generic;  
using System.Threading.Tasks;  
  
namespace Amazon.DocSamples.S3  
{  
    class TempFederatedCredentialsTest  
    {  
        private const string bucketName = "**** bucket name ****";  
        // Specify your bucket region (an example region is shown).  
        private static readonly RegionEndpoint bucketRegion =  
        RegionEndpoint.USWest2;  
        private static IAmazonS3 client;  
  
        public static void Main()  
        {  
            ListObjectsAsync().Wait();  
        }  
  
        private static async Task ListObjectsAsync()  
        {  
            try  
            {  
                Console.WriteLine("Listing objects stored in a bucket");  
                // Credentials use the default AWS SDK for .NET credential search  
                chain.  

```

```
// On local development machines, this is your default profile.
SessionAWSCredentials tempCredentials =
    await GetTemporaryFederatedCredentialsAsync();

// Create a client by providing temporary security credentials.
using (client = new AmazonS3Client(bucketRegion))
{
    ListObjectsRequest listObjectRequest = new
ListObjectsRequest();
    listObjectRequest.BucketName = bucketName;

    ListObjectsResponse response = await
client.ListObjectsAsync(listObjectRequest);
    List<S3Object> objects = response.S3Objects;
    Console.WriteLine("Object count = {0}", objects.Count);

    Console.WriteLine("Press any key to continue...");
    Console.ReadKey();
}
}
catch (AmazonS3Exception e)
{
    Console.WriteLine("Error encountered ***. Message:'{0}' when
writing an object", e.Message);
}
catch (Exception e)
{
    Console.WriteLine("Unknown encountered on server. Message:'{0}'
when writing an object", e.Message);
}
}

private static async Task<SessionAWSCredentials>
GetTemporaryFederatedCredentialsAsync()
{
    AmazonSecurityTokenServiceConfig config = new
AmazonSecurityTokenServiceConfig();
    AmazonSecurityTokenServiceClient stsClient =
        new AmazonSecurityTokenServiceClient(
            config);

    GetFederationTokenRequest federationTokenRequest =
        new GetFederationTokenRequest();
    federationTokenRequest.DurationSeconds = 7200;
```

```

        federationTokenRequest.Name = "User1";
        federationTokenRequest.Policy = @"{
            ""Statement"":
            [
                {
                    ""Sid"":""Stmnt1311212314284"",
                    ""Action"":[""s3:ListBucket""],
                    ""Effect"":""Allow"",
                    ""Resource"":""arn:aws:s3:::" + bucketName + @""
                }
            ]
        }
";

        GetFederationTokenResponse federationTokenResponse =
            await
            stsClient.GetFederationTokenAsync(federationTokenRequest);
        Credentials credentials = federationTokenResponse.Credentials;

        SessionAWSCredentials sessionCredentials =
            new SessionAWSCredentials(credentials.AccessKeyId,
                                      credentials.SecretAccessKey,
                                      credentials.SessionToken);

        return sessionCredentials;
    }
}

```

PHP

Questo argomento spiega come utilizzare le classi della versione 3 per richiedere credenziali di sicurezza temporanee per utenti e applicazioni federati e utilizzarle per accedere alle risorse archiviate in Amazon S3. AWS SDK per PHP Per ulteriori informazioni sull'API AWS SDK for Ruby, consulta SDK for Ruby - [AWS Version 2](#).

Puoi fornire credenziali di sicurezza temporanee agli utenti e alle applicazioni federati in modo che possano inviare richieste autenticate per accedere alle tue risorse. AWS Quando si richiedono le credenziali temporanee, è necessario fornire un nome utente e una policy IAM che descrive le autorizzazioni a livello di risorsa che si desidera concedere. Queste credenziali scadono alla scadenza della durata della sessione. Per default, la sessione dura un'ora. In fase di richiesta delle credenziali di sicurezza temporanee per utenti federati e applicazioni, è possibile impostare

in modo esplicito un valore di durata diverso. Per ulteriori informazioni sulle credenziali di sicurezza temporanee, consulta la sezione relativa alle [credenziali di sicurezza temporanee](#) nella Guida per l'utente IAM. Per ulteriori informazioni sulla specifica delle credenziali di sicurezza temporanee alle applicazioni e agli utenti federati, consulta [Esecuzione di richieste](#).

Per una maggiore sicurezza nella richiesta di credenziali di sicurezza temporanee per utenti federati e applicazioni, consigliamo di utilizzare un utente IAM dedicato che abbia solo le autorizzazioni di accesso necessarie. L'utente temporaneo che si crea non può mai avere autorizzazioni più ampie rispetto all'utente IAM che ha richiesto le credenziali di sicurezza temporanee. Per ulteriori informazioni, consulta l'articolo sulla federazione delle identità Web in [Domande frequenti di AWS Identity and Access Management](#).

Per ulteriori informazioni sull'API AWS SDK for Ruby, vai a SDK for Ruby - [AWS Version 2](#).

Example

L'esempio PHP seguente elenca le chiavi incluse nel bucket specificato. Nell'esempio, è necessario ottenere le credenziali di sicurezza temporanee per l'utente federato (User1) per una sessione della durata di un'ora. Poi si utilizzano le credenziali di sicurezza temporanee per inviare richieste autenticate ad Amazon S3.

Per una maggiore sicurezza quando si effettua la richiesta di credenziali temporanee per altri soggetti, si utilizzano le credenziali di sicurezza di un utente IAM che abbia le autorizzazioni per richiedere credenziali di sicurezza temporanee. Per concedere all'utente IAM solo le autorizzazioni minime specifiche delle applicazioni per l'utente federato, puoi limitare anche le autorizzazioni di accesso dell'utente IAM. In questo esempio sono elencati solo gli oggetti di un bucket specifico. Crea un utente IAM con la seguente policy collegata:

```
{
  "Statement": [{
    "Action": ["s3:ListBucket",
      "sts:GetFederationToken*"],
    "Effect": "Allow",
    "Resource": "*"
  }]
}
```

La policy consente all'utente IAM di richiedere credenziali di sicurezza temporanee e autorizzazioni di accesso solo per elencare le risorse. AWS Per ulteriori informazioni su come

creare un utente IAM, consulta [Creazione del primo utente IAM e del gruppo di amministratori](#) nella Guida per l'utente di IAM.

Puoi ora utilizzare le credenziali di sicurezza dell'utente IAM per provare l'esempio seguente. Nell'esempio viene inviata una richiesta autenticata ad Amazon S3 mediante credenziali di sicurezza temporanee. Durante la richiesta di credenziali di sicurezza temporanee per l'utente federato (User1), nell'esempio è specificata la seguente policy, con accesso limitato agli oggetti dell'elenco di un bucket specifico. Aggiornare la policy con il nome del bucket.

```
{
  "Statement": [
    {
      "Sid": "1",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::YourBucketName"
    }
  ]
}
```

Nell'esempio che segue, quando specifichi la risorsa della policy, è necessario sostituire YourBucketName con il nome del tuo bucket.:

```
require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$bucket = '*** Your Bucket Name ***';

// In real applications, the following code is part of your trusted code. It has
// the security credentials that you use to obtain temporary security credentials.
$sts = new StsClient([
    'version' => 'latest',
    'region' => 'us-east-1'
]);

// Fetch the federated credentials.
$sessionToken = $sts->getFederationToken([
    'Name' => 'User1',
    'DurationSeconds' => '3600',
```

```

        'Policy'          => json_encode([
            'Statement' => [
                'Sid'          => 'randomstatementid' . time(),
                'Action'       => ['s3:ListBucket'],
                'Effect'       => 'Allow',
                'Resource'     => 'arn:aws:s3:::' . $bucket
            ]
        ])
    });

// The following will be part of your less trusted code. You provide temporary
// security credentials so the code can send authenticated requests to Amazon S3.

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
    'credentials' => [
        'key' => $sessionToken['Credentials']['AccessKeyId'],
        'secret' => $sessionToken['Credentials']['SecretAccessKey'],
        'token' => $sessionToken['Credentials']['SessionToken']
    ]
]);

try {
    $result = $s3->listObjects([
        'Bucket' => $bucket
    ]);
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}

```

Ruby

Puoi fornire credenziali di sicurezza temporanee per gli utenti e le applicazioni federati in modo che possano inviare richieste autenticate per accedere alle tue risorse. AWS Quando si richiedono tali credenziali temporanee a un servizio IAM, è necessario fornire un nome utente e una policy IAM che descrive le autorizzazioni a livello di risorsa che si desidera concedere. Per default, la sessione dura un'ora. Tuttavia, se si sta effettuando la richiesta di credenziali temporanee mediante credenziali utente IAM, è possibile definire espressamente un valore diverso per la durata quando si richiedono credenziali di sicurezza temporanee per utenti federati e applicazioni.

Per ulteriori informazioni sulle credenziali di sicurezza temporanee per le applicazioni e gli utenti federati, consulta [Esecuzione di richieste](#).

Note

Per una maggiore sicurezza, quando richiedi credenziali di sicurezza temporanee per utenti federati e applicazioni, potrebbe essere preferibile usare un utente IAM dedicato che abbia solo le autorizzazioni di accesso necessarie. L'utente temporaneo che si crea non può mai avere autorizzazioni più ampie rispetto all'utente IAM che ha richiesto le credenziali di sicurezza temporanee. Per ulteriori informazioni, consulta [FAQ AWS Identity and Access Management](#).

Example

Nel seguente esempio di codice Ruby consente a un utente con una serie limitata di autorizzazione per visualizzare le chiavi nel bucket specificato.

```
# Prerequisites:
# - An existing Amazon S3 bucket.

require 'aws-sdk-s3'
require 'aws-sdk-iam'
require 'json'

# Checks to see whether a user exists in IAM; otherwise,
# creates the user.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Aws::IAM::Types::User] The existing or new user.
# @example
#   iam = Aws::IAM::Client.new(region: 'us-west-2')
#   user = get_user(iam, 'my-user')
#   exit 1 unless user.user_name
#   puts "User's name: #{user.user_name}"
def get_user(iam, user_name)
  puts "Checking for a user with the name '#{user_name}'..."
  response = iam.get_user(user_name: user_name)
  puts "A user with the name '#{user_name}' already exists."
  response.user
# If the user doesn't exist, create them.
```



```

rescue Aws::IAM::Errors::NoSuchEntity
  puts "A user with the name '#{user_name}' doesn't exist. Creating this user..."
  response = iam.create_user(user_name: user_name)
  iam.wait_until(:user_exists, user_name: user_name)
  puts "Created user with the name '#{user_name}'."
  response.user
rescue StandardError => e
  puts "Error while accessing or creating the user named '#{user_name}':
    #{e.message}"
end

# Gets temporary AWS credentials for an IAM user with the specified permissions.
#
# @param sts [Aws::STS::Client] An initialized AWS STS client.
# @param duration_seconds [Integer] The number of seconds for valid credentials.
# @param user_name [String] The user's name.
# @param policy [Hash] The access policy.
# @return [Aws::STS::Types::Credentials] AWS credentials for API authentication.
# @example
#   sts = Aws::STS::Client.new(region: 'us-west-2')
#   credentials = get_temporary_credentials(sts, duration_seconds, user_name,
#     {
#       'Version' => '2012-10-17',
#       'Statement' => [
#         'Sid' => 'Stmt1',
#         'Effect' => 'Allow',
#         'Action' => 's3:ListBucket',
#         'Resource' => 'arn:aws:s3:::amzn-s3-demo-bucket'
#       ]
#     }
#   )
#   exit 1 unless credentials.access_key_id
#   puts "Access key ID: #{credentials.access_key_id}"
def get_temporary_credentials(sts, duration_seconds, user_name, policy)
  response = sts.get_federation_token(
    duration_seconds: duration_seconds,
    name: user_name,
    policy: policy.to_json
  )
  response.credentials
rescue StandardError => e
  puts "Error while getting federation token: #{e.message}"
end

```

```

# Lists the keys and ETags for the objects in an Amazon S3 bucket.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if the objects were listed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_objects_in_bucket?(s3_client, 'amzn-s3-demo-bucket')
def list_objects_in_bucket?(s3_client, bucket_name)
  puts "Accessing the contents of the bucket named '#{bucket_name}'..."
  response = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if response.count.positive?
    puts "Contents of the bucket named '#{bucket_name}' (first 50 objects):"
    puts 'Name => ETag'
    response.contents.each do |obj|
      puts "#{obj.key} => #{obj.etag}"
    end
  else
    puts "No objects in the bucket named '#{bucket_name}'."
  end
  true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
end

# Example usage:
def run_me
  region = "us-west-2"
  user_name = "my-user"
  bucket_name = "amzn-s3-demo-bucket"

  iam = Aws::IAM::Client.new(region: region)
  user = get_user(iam, user_name)

  exit 1 unless user.user_name

  puts "User's name: #{user.user_name}"
  sts = Aws::STS::Client.new(region: region)
  credentials = get_temporary_credentials(sts, 3600, user_name,
                                          {

```

```

        'Version' => '2012-10-17',
        'Statement' => [
            'Sid' => 'Stmt1',
            'Effect' => 'Allow',
            'Action' => 's3:ListBucket',
            'Resource' =>
                "arn:aws:s3:::#{bucket_name}"
        ]
    })

    exit 1 unless credentials.access_key_id

    puts "Access key ID: #{credentials.access_key_id}"
    s3_client = Aws::S3::Client.new(region: region, credentials: credentials)

    exit 1 unless list_objects_in_bucket?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__

```

Esecuzione di richieste con l'utilizzo di API REST

Questa sezione contiene informazioni su come effettuare richieste agli endpoint Amazon S3 utilizzando l'API REST. Per un elenco di endpoint di Amazon S3, consultare la sezione relativa a [regioni ed endpoint](#) nella Riferimenti generali di AWS.

Costruzione di nomi host S3 per richieste API REST

Gli endpoint Amazon S3 seguono la struttura riportata di seguito:

```
s3.Region.amazonaws.com
```

Gli endpoint degli Punti di accesso Amazon S3 e gli endpoint dual-stack seguono la struttura standard:

- Access point Amazon S -s3-accesspoint.*Region*.amazonaws.com
- Dual-stack - s3.dualstack.*Region*.amazonaws.com

Per un elenco completo degli endpoint e delle regioni Amazon S3, consultare la sezione relativa a [endpoint e quote di Amazon S3](#) nella Riferimenti generali di Amazon Web Services.

Richieste in stile hosting virtuale e in stile percorso

Quando si effettua una richiesta utilizzando l'API REST, è possibile utilizzare URI in stile hosting virtuale o in stile percorso per gli endpoint Amazon S3. Per ulteriori informazioni, consulta [Path-style le richieste](#).

Example Richiesta in stile hosting virtuale

Qui di seguito è riportato un esempio di richiesta in stile hosting virtuale per eliminare il file `puppy.jpg` dal bucket denominato `examplebucket` nella regione Stati Uniti occidentali (Oregon). Per ulteriori informazioni sulle richieste in stile host virtuale, vedi [Path-style richieste](#).

```
DELETE /puppy.jpg HTTP/1.1
Host: examplebucket.s3.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Example Path-style richiesta

Di seguito è riportato un esempio di una versione in stile percorso della stessa richiesta.

```
DELETE /examplebucket/puppy.jpg HTTP/1.1
Host: s3.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Attualmente, Amazon S3 supporta sia gli URL in stile hosting virtuale che quelli in stile percorso in tutte le Regioni AWS. Tuttavia, gli URL in stile percorso non saranno più disponibili in futuro. Per ulteriori informazioni, consulta la seguente nota importante.

Per ulteriori informazioni sulle richieste in stile percorso, vedi [Path-style richieste](#).

Important

Aggiornamento (23 settembre 2020): per assicurare che i clienti abbiano il tempo necessario per eseguire la transizione agli URL in stile hosting virtuale, abbiamo deciso di posticipare

l'obsolescenza degli URL in stile percorso. Per ulteriori informazioni, consulta [Piano di obsolescenza del percorso Amazon S3 - Il resto della storia](#) nel Blog AWS News.

Esecuzione di richieste a endpoint Dual-Stack utilizzando l'API REST

Quando si utilizza l'API REST, è possibile accedere direttamente a un endpoint dual-stack utilizzando un nome di endpoint (URI) in stile hosting virtuale o in stile percorso. Tutti i nomi di endpoint dual-stack Amazon S3 includono la regione nel nome. A differenza degli IPv4-only endpoint standard, sia gli endpoint in stile host virtuale che quelli in stile path utilizzano nomi di endpoint specifici per regione.

Example Richiesta con endpoint dual-stack in stile hosting virtuale

È possibile utilizzare un endpoint in stile hosting virtuale nella richiesta REST come illustrato nell'esempio seguente che recupera l'oggetto puppy.jpg dal bucket denominato examplebucket nella regione Stati Uniti occidentali (Oregon).

```
GET /puppy.jpg HTTP/1.1
Host: examplebucket.s3.dualstack.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Example Path-style richiesta di endpoint dual-stack

Altrimenti, è possibile utilizzare un endpoint in stile percorso nella richiesta come illustrato nell'esempio seguente.

```
GET /examplebucket/puppy.jpg HTTP/1.1
Host: s3.dualstack.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Per ulteriori informazioni sugli endpoint dual-stack, consulta [Utilizzo degli endpoint dual-stack Amazon S3](#).

Per ulteriori informazioni su come effettuare richieste utilizzando l'API REST, consulta i seguenti argomenti.

Argomenti

- [Reindirizzamento delle richieste e API REST](#)
- [Instradamento della richiesta](#)

Reindirizzamento delle richieste e API REST

Argomenti

- [Reindirizzamenti e utenti-agenti HTTP](#)
- [Reindirizzamenti e 100-continue](#)
- [Esempio di reindirizzamento](#)

Questa sezione descrive come gestire i reindirizzamenti HTTP tramite l'API REST di Amazon S3. Per informazioni generali sui reindirizzamenti di Amazon S3, consulta [Esecuzione di richieste](#) nella Documentazione di riferimento delle API di Amazon Simple Storage Service.

Reindirizzamenti e utenti-agenti HTTP

I programmi che utilizzano l'API REST di Amazon S3 devono gestire i reindirizzamenti a livello di applicazione o a livello di HTTP. Molte librerie di client e utenti-agenti HTTP possono essere configurate per gestire automaticamente i reindirizzamenti in modo corretto; tuttavia, in molti altri casi, le implementazioni sono incorrette o incomplete.

Prima di affidarsi a una libreria per soddisfare il requisito di reindirizzamento, testare i casi riportati di seguito:

- Verificare che la richiesta reindirizzata (la seconda richiesta dopo la ricezione del reindirizzamento) includa tutte le intestazioni della richiesta HTTP, comprese quelle standard relative, ad esempio, ad autorizzazione e data.
- Verificare il corretto funzionamento dei reindirizzamenti non GET, come PUT e DELETE.
- Verificare che le richieste PUT di grandi dimensioni seguano correttamente i reindirizzamenti.
- Se la risposta 100-continue impiega molto tempo ad arrivare, verificare che le richieste PUT seguano correttamente i reindirizzamenti.

Se il metodo di richiesta HTTP non è GET o HEAD, gli utenti-agenti HTTP rigorosamente conformi allo standard RFC 2616 potrebbero richiedere una conferma esplicita prima di seguire un

reindirizzamento. In genere è sicuro seguire i reindirizzamenti generati automaticamente da Amazon S3, perché il sistema li emette solo verso host interni al dominio `amazonaws.com` e l'effetto della richiesta reindirizzata è uguale a quello della richiesta originale.

Reindirizzamenti e 100-continue

Configurare l'applicazione per l'utilizzo di 100-continue per le operazioni PUT consente di semplificare la gestione dei reindirizzamenti, migliorare l'efficienza ed evitare i costi che comporta inviare due volte un corpo della richiesta reindirizzato. Con l'utilizzo di 100-continue, l'applicazione non invia il corpo della richiesta finché non riceve una conferma. Se, in base alle intestazioni, il messaggio viene rifiutato, il corpo del messaggio non viene inviato. Per ulteriori informazioni su 100-continue, consulta [RFC 2616 sezione 8.2.3](#)

Note

Secondo lo standard RFC 2616, quando si utilizza `Expect: Continue` con un server HTTP sconosciuto, non bisogna attendere un periodo di tempo indefinito prima inviare il corpo della richiesta, perché alcuni server HTTP non riconoscono 100-continue. Tuttavia, Amazon S3 è in grado di riconoscere se la richiesta contiene un codice `Expect: Continue` e risponderà con uno stato 100-continue provvisorio o con un codice di stato definitivo. Inoltre, dopo aver ricevuto il via libera provvisorio per 100-continue, non si verificheranno errori di reindirizzamento: in questo modo non si riceverà una risposta di reindirizzamento mentre è ancora in corso la scrittura del corpo della richiesta.

Esempio di reindirizzamento

In questa sezione è riportato un esempio di interazione fra client e server con utilizzo di reindirizzamenti HTTP e 100-continue.

Di seguito è riportato un esempio di operazione PUT nel bucket `quotes.s3.amazonaws.com`.

```
PUT /nelson.txt HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000

Content-Length: 6
Expect: 100-continue
```

Amazon S3 restituisce quanto segue:

```
HTTP/1.1 307 Temporary Redirect
Location: http://quotes.s3-4c25d83b.amazonaws.com/nelson.txt?rk=8d47490b
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Mon, 15 Oct 2007 22:18:46 GMT

Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
  <Message>Please re-send this request to the
    specified temporary endpoint. Continue to use the
    original request endpoint for future requests.
  </Message>
  <Endpoint>quotes.s3-4c25d83b.amazonaws.com</Endpoint>
  <Bucket>quotes</Bucket>
</Error>
```

Il client segue la risposta di reindirizzamento ed emette una nuova richiesta per l'endpoint temporaneo `quotes.s3-4c25d83b.amazonaws.com`.

```
PUT /nelson.txt?rk=8d47490b HTTP/1.1
Host: quotes.s3-4c25d83b.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000

Content-Length: 6
Expect: 100-continue
```

Amazon S3 restituisce un 100-continue indicando al client di procedere con l'invio del corpo della richiesta.

```
HTTP/1.1 100 Continue
```

Il client invia il corpo della richiesta.

```
ha ha\n
```

Amazon S3 restituisce la risposta definitiva.


```
HTTP/1.1 200 OK
Date: Mon, 15 Oct 2007 22:18:48 GMT

ETag: "a2c8d6b872054293afd41061e93bc289"
Content-Length: 0
Server: AmazonS3
```

Instradamento della richiesta

I programmi che effettuano richieste su bucket creati utilizzando l'[CreateBucket](#) API che includono un [CreateBucketConfiguration](#) must support per i reindirizzamenti. Inoltre, è possibile che si verifichino problemi con alcuni client che non rispettano i valori TTL DNS.

Questa sezione descrive l'instradamento e i problemi del server DNS da tenere in considerazione durante la progettazione del servizio o dell'applicazione da utilizzare con Amazon S3.

Reindirizzamento delle richieste e API REST

Amazon S3 utilizza Domain Name System (DNS) per instradare le richieste a strutture che possano elaborarle. Il sistema funziona in modo efficace, ma potrebbero verificarsi errori di instradamento temporaneo. Se una richiesta arriva nella posizione Amazon S3 sbagliata, Amazon S3 risponde con un reindirizzamento temporaneo che indica al richiedente di rinviare la richiesta a un altro endpoint. Se la richiesta è formulata in modo errato, Amazon S3 utilizza i reindirizzamenti permanenti per fornire indicazioni su come eseguire la richiesta correttamente.

Important

Per utilizzare questa funzionalità, è necessario essere in possesso di un'applicazione che possa gestire le risposte di reindirizzamento di Amazon S3. L'unica eccezione riguarda le applicazioni che operano esclusivamente con bucket creati senza `<CreateBucketConfiguration>`. Per ulteriori informazioni sui vincoli di posizione, consulta [Accedere ed elencare un bucket Amazon S3](#).

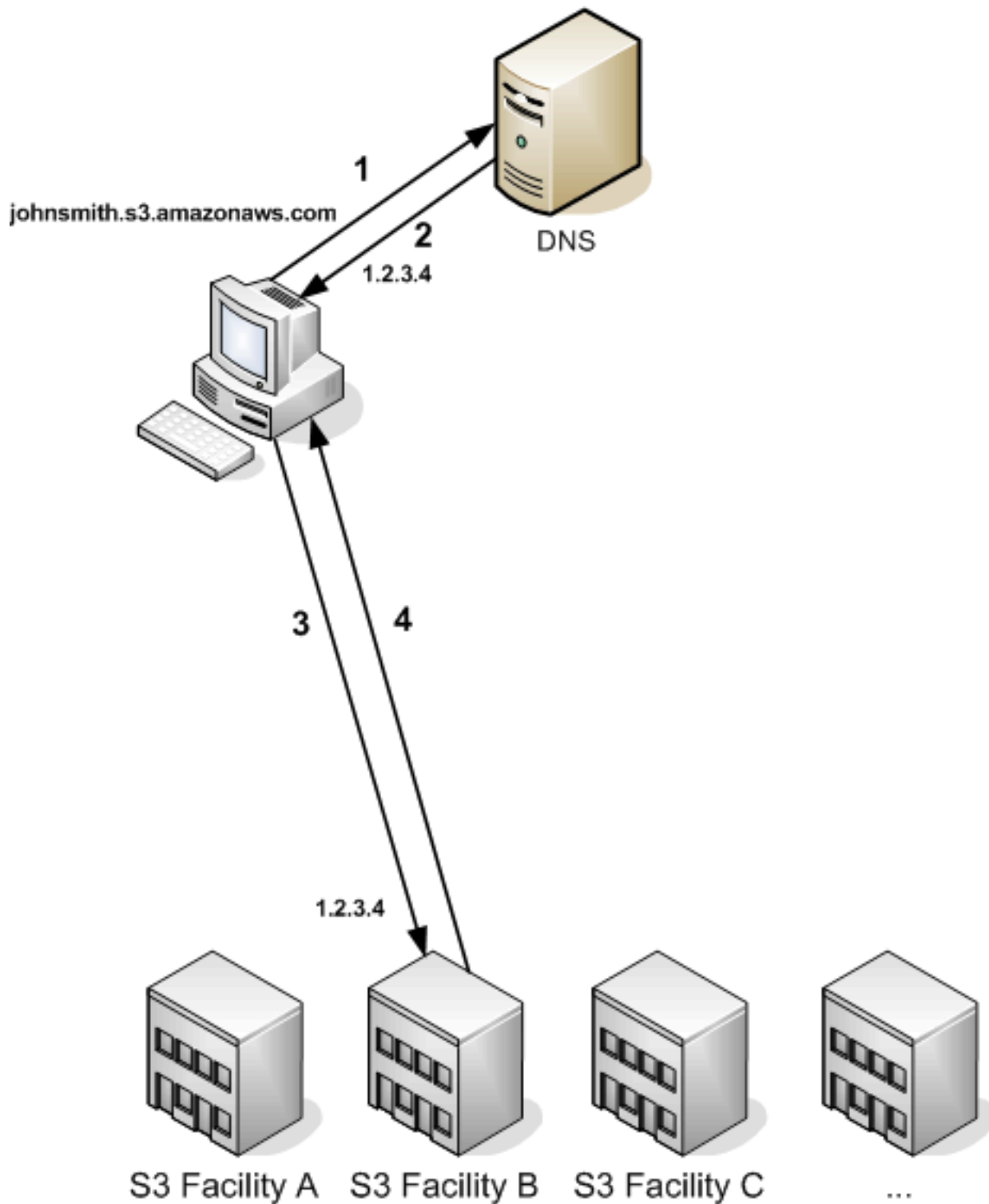
Per tutte le regioni lanciate dopo il 20 marzo 2019, se una richiesta arriva alla posizione Amazon S3 errata, Amazon S3 restituisce un errore HTTP 400 - Richiesta non valida. Per ulteriori informazioni sull'attivazione o la disabilitazione di un Regione AWS, consulta [Regioni AWS Endpoints in](#). Riferimenti generali di AWS

Argomenti

- [Instradamento DNS](#)
- [Reindirizzamento temporaneo delle richieste](#)
- [Reindirizzamento permanente delle richieste](#)
- [Esempi di reindirizzamento delle richieste](#)

Instradamento DNS

L'instradamento DNS indirizza le richieste alle strutture Amazon S3 appropriate. La figura e la procedura seguenti illustrano un esempio di instradamento DNS.



Passaggi della richiesta di instradamento DNS

1. Il client effettua una richiesta DNS per l'archiviazione di un oggetto su Amazon S3.

2. Il client riceve uno o più indirizzi IP per le strutture che possono elaborare la richiesta. In questo esempio, l'indirizzo IP è per la struttura B.
3. Il client effettua una richiesta alla struttura B di Amazon S3.
4. La struttura B restituisce una copia dell'oggetto al client.

Reindirizzamento temporaneo delle richieste

Il reindirizzamento temporaneo è un tipo di risposta di errore che segnala al richiedente la necessità di rinviare la richiesta a un altro endpoint. Data la natura distribuita di Amazon S3, le richieste possono essere temporaneamente instradate alla struttura sbagliata. Questo si verifica con maggiore probabilità immediatamente dopo la creazione o l'eliminazione dei bucket.

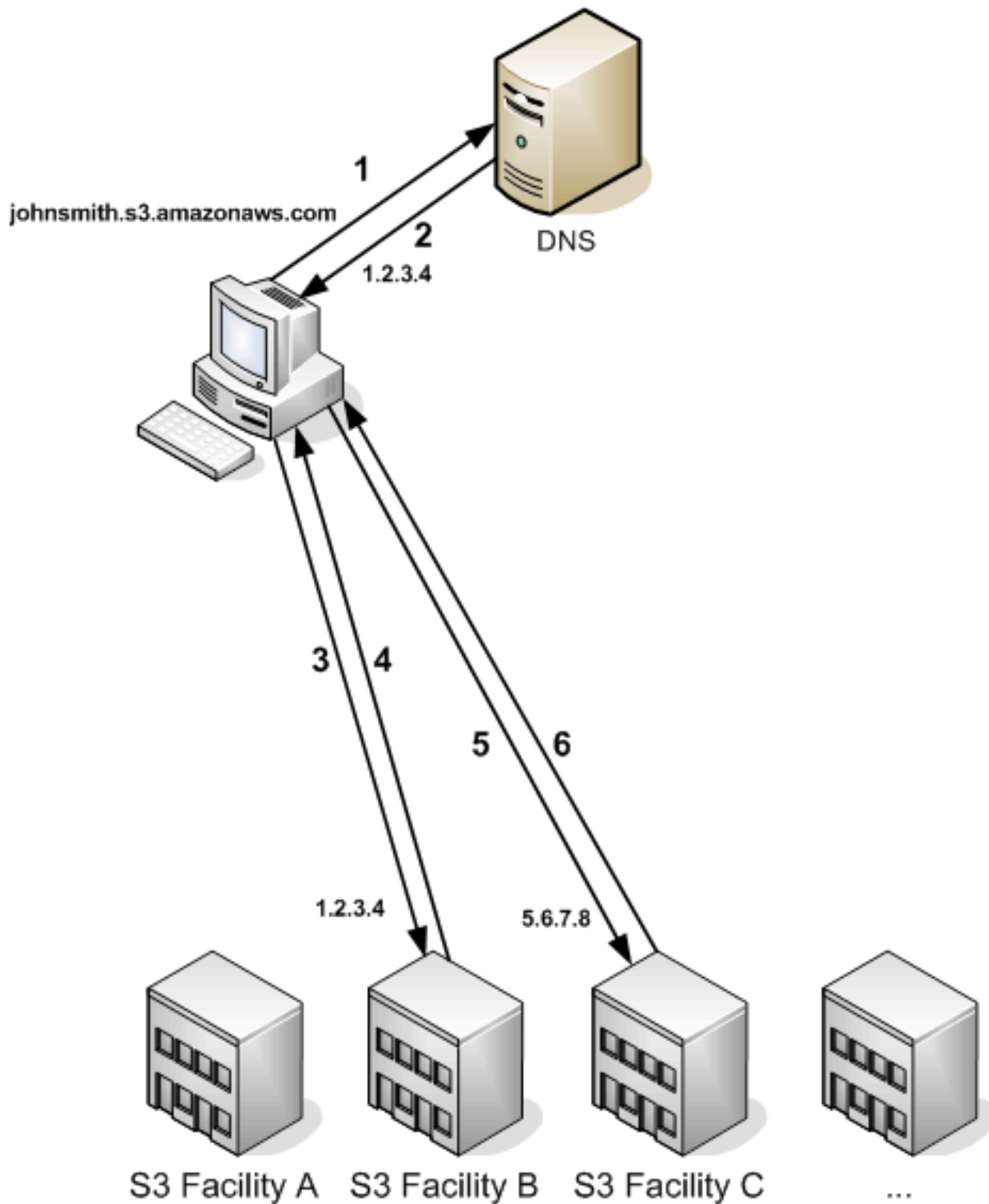
Ad esempio, se si crea un nuovo bucket e si effettua immediatamente una richiesta al bucket, si potrebbe ricevere un reindirizzamento temporaneo, a seconda del vincolo di posizione del bucket. Se hai creato il bucket negli Stati Uniti orientali (Virginia settentrionale) Regione AWS, non vedrai il reindirizzamento perché questo è anche l'endpoint Amazon S3 predefinito.

Tuttavia, se il bucket viene creato in qualsiasi altra regione, tutte le richieste al bucket arrivano all'endpoint predefinito durante la propagazione della voce DNS del bucket. L'endpoint predefinito reindirizza la richiesta all'endpoint corretto con una risposta HTTP 302. I reindirizzamenti temporanei contengono un URI per la struttura corretta, che può essere utilizzato per rinviare immediatamente la richiesta.

Important

Non riutilizzare un endpoint fornito da una risposta di reindirizzamento precedente. Potrebbe sembrare funzionante (anche per lunghi periodi di tempo), ma potrebbe generare risultati imprevedibili e infine dare esito negativo senza preavviso.

La figura e la procedura seguenti illustrano un esempio di reindirizzamento temporaneo.



Passaggi di reindirizzamento temporaneo delle richieste

1. Il client effettua una richiesta DNS per l'archiviazione di un oggetto su Amazon S3.
2. Il client riceve uno o più indirizzi IP per le strutture che possono elaborare la richiesta.

3. Il client effettua una richiesta alla struttura B di Amazon S3.
4. La struttura B restituisce un reindirizzamento che indica che l'oggetto è disponibile dalla posizione C.
5. Il client rinvia la richiesta alla struttura C.
6. La struttura C restituisce una copia dell'oggetto.

Reindirizzamento permanente delle richieste

Un reindirizzamento permanente indica che la richiesta è stata rivolta a una risorsa in modo non appropriato. Ad esempio, i reindirizzamenti permanenti si verificano in caso di utilizzo di una richiesta in stile percorso per accedere a un bucket creato utilizzando `<CreateBucketConfiguration>`. Per ulteriori informazioni, consulta [Accedere ed elencare un bucket Amazon S3](#).

Per agevolare l'individuazione di questi errori durante lo sviluppo, questo tipo di reindirizzamento non contiene un'intestazione HTTP di posizione che permette di seguire automaticamente la richiesta verso la posizione corretta. Consulta il documento di errore XML risultante come supporto per utilizzare l'endpoint Amazon S3 corretto.

Esempi di reindirizzamento delle richieste

I seguenti sono esempi di risposte di reindirizzamento temporaneo delle richieste.

Risposta di reindirizzamento temporaneo API REST

```
HTTP/1.1 307 Temporary Redirect
Location: http://awsexamplebucket1.s3-gztb4pa9sq.amazonaws.com/photos/puppy.jpg?rk=e2c69a31
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Fri, 12 Oct 2007 01:12:56 GMT
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
  <Message>Please re-send this request to the specified temporary endpoint. Continue to use the original request endpoint for future requests.</Message>
  <Endpoint>awsexamplebucket1.s3-gztb4pa9sq.amazonaws.com</Endpoint>
</Error>
```

Risposta di reindirizzamento temporaneo API SOAP

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.TemporaryRedirect</Faultcode>
    <Faultstring>Please re-send this request to the specified temporary endpoint.
    Continue to use the original request endpoint for future requests.</Faultstring>
    <Detail>
      <Bucket>images</Bucket>
      <Endpoint>s3-gztlb4pa9sq.amazonaws.com</Endpoint>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

Considerazioni sul sistema DNS

Uno dei requisiti di progettazione di Amazon S3 è una disponibilità estremamente alta. Per rispondere a questo requisito, è possibile aggiornare gli indirizzi IP associati all'endpoint Amazon S3 in DNS, in base alle necessità. Queste modifiche si riflettono automaticamente nei client di breve durata, ma non in alcuni client a lunga durata. Long-lived i clienti dovranno intraprendere azioni speciali per risolvere periodicamente l'endpoint Amazon S3 per beneficiare di queste modifiche. Per ulteriori informazioni sulle macchine virtuali (VM), consultare:

- Per Java, la JVM di Sun memorizza nella cache le ricerche DNS per sempre per impostazione predefinita; vai alla sezione "InetAddressCaching» della InetAddress documentazione [per informazioni su come modificare questo](#) comportamento.
- Per PHP, la VM PHP persistente che esegue le configurazioni di distribuzione più diffuse memorizza nella cache le ricerche DNS fino al riavvio della VM. Vai alla documentazione per scaricare PHP. [HostByName](#)

Sviluppo con Amazon S3 utilizzando AWS CLI

I AWS CLI comandi Amazon S3 sono organizzati in diversi set in AWS CLI Command Reference e ognuno ha i propri comandi disponibili. Se non trovi il comando che stai cercando in un set, seleziona uno degli altri set. I diversi set sono i seguenti:

- **s3**— Descrive i comandi di alto livello per lavorare con oggetti e bucket. Questi includono le azioni di copia, elenco e spostamento. Per un elenco completo dei comandi di questo set, vedere [s3](#) nel AWS CLI Command Reference.
- **s3api**— Descrive i comandi di basso livello che gestiscono le risorse S3 come bucket, oggetti, sessioni e caricamenti multipart. Questi comandi corrispondono alle operazioni dell'API Amazon S3. Per un elenco completo dei comandi CLI in questo set, consulta [s3api](#) nel Command Reference. AWS CLI
- **s3control**— Descrive i comandi di basso livello che gestiscono tutte le altre risorse S3 come Access Grants, Storage Lens groups e i bucket Amazon S3 on Outposts. Questi comandi corrispondono alle operazioni dell'API Amazon S3 Control. Per un elenco completo dei comandi CLI in questo set, consulta [s3control](#) nel Command Reference. AWS CLI

In caso di problemi con la configurazione di AWS CLI, consulta [How do I resolve the error «Unable to locate credentials» in Amazon S3 nel Knowledge Center](#). AWS re:Post

Note

I servizi in AWS, come Amazon S3, richiedono l'immissione delle credenziali quando si accede ad essi. Il servizio può quindi stabilire se si dispone delle autorizzazioni per accedere alle risorse del servizio stesso. Per la console è necessaria la password. Puoi creare chiavi di accesso per accedere Account AWS all'API AWS CLI or. Tuttavia, non accedere AWS utilizzando le credenziali del tuo Account AWS. Ti consigliamo invece di utilizzare AWS Identity and Access Management (IAM). Crea un utente IAM, aggiungilo a un gruppo IAM con autorizzazioni amministrative, quindi concedi le autorizzazioni amministrative all'utente IAM creato. Puoi quindi accedere ad AWS usando un URL speciale e le credenziali di tale utente IAM. Per istruzioni, consulta [Creazione del primo utente IAM e del gruppo di amministratori](#) nella Guida per l'utente di IAM.

Scopri di più su AWS CLI

Per saperne di più su AWS, consulta le seguenti risorse:

- [AWS Command Line Interface](#)
- [AWS Command Line Interface Guida per l'utente per la versione 2](#)

Sviluppo con Amazon S3 utilizzando AWS SDK

AWS I kit di sviluppo software (SDK) sono disponibili per molti linguaggi di programmazione diffusi. Ogni SDK fornisce un'API, esempi di codice e documentazione che facilitano agli sviluppatori la creazione di applicazioni nel loro linguaggio preferito.

Note

Puoi utilizzarli AWS Amplify per lo sviluppo completo end-to-end di app web e mobili. Amplify Storage integra perfettamente le funzionalità di archiviazione e gestione dei file nelle app frontend web e mobili, basate su Amazon S3. Per ulteriori informazioni, consulta [Storage](#) nella guida per l'utente di Amplify.

Documentazione sugli SDK	Esempi di codice
AWS SDK per C++	AWS SDK per C++ esempi di codice
AWS CLI	AWS CLI esempi di codice
AWS SDK per Go	AWS SDK per Go esempi di codice
AWS SDK per Java	AWS SDK per Java esempi di codice
AWS SDK per JavaScript	AWS SDK per JavaScript esempi di codice
AWS SDK per Kotlin	AWS SDK per Kotlin esempi di codice
AWS SDK per .NET	AWS SDK per .NET esempi di codice
AWS SDK per PHP	AWS SDK per PHP esempi di codice

Documentazione sugli SDK	Esempi di codice
AWS Strumenti per PowerShell	Strumenti per esempi di PowerShell codice
AWS SDK per Python (Boto3)	AWS SDK per Python (Boto3) esempi di codice
AWS SDK per Ruby	AWS SDK per Ruby esempi di codice
AWS SDK per Rust	AWS SDK per Rust esempi di codice
AWS SDK per SAP ABAP	AWS SDK per SAP ABAP esempi di codice
AWS SDK per Swift	AWS SDK per Swift esempi di codice

Per esempi specifici consulta [Esempi di codice per l'utilizzo di Amazon S3 AWS SDK](#).

Interfacce di programmazione SDK

Ogni AWS SDK fornisce una o più interfacce programmatiche per lavorare con Amazon S3. Ogni SDK fornisce un'interfaccia di basso livello per Amazon S3, con metodi che assomigliano molto alle operazioni API. Alcuni SDK forniscono interfacce di alto livello per Amazon S3, che sono astrazioni destinate a semplificare i casi d'uso più comuni.

Ad esempio, quando esegui un caricamento in più parti utilizzando le operazioni API di basso livello, usi un'operazione per avviare il caricamento, un'altra operazione per caricare le parti e un'operazione finale per completare il caricamento. Un'operazione API di caricamento multipart di alto livello consente di eseguire tutte le operazioni necessarie per il caricamento in una singola chiamata API. Per esempi, consulta [Caricare un oggetto mediante il caricamento in più parti](#) nella Amazon S3 User Guide.

Low-level Le operazioni API consentono un maggiore controllo sul caricamento. Ti consigliamo di utilizzare le operazioni API di basso livello se devi mettere in pausa e riprendere i caricamenti, variare le dimensioni delle parti durante il caricamento o iniziare i caricamenti quando non conosci in anticipo la dimensione dei dati.

Specificare la versione della firma nell'autenticazione della richiesta

Nella maggior parte dei casi, Amazon S3 supporta solo AWS la versione 4 di Signature. Regioni AWS In alcune delle versioni precedenti Regioni AWS, Amazon S3 supporta sia la versione 4 di Signature

che la versione 2 di Signature. Comunque, Signature Version 2 sta per essere disattivato perché obsoleto. Per ulteriori informazioni sulla fine del supporto per Signature Version 2, consulta [AWS Signature Version 2 disattivata \(obsoleta\) per Amazon S3](#).

Per un elenco di tutte le regioni Amazon S3 e delle versioni di firma supportate, consulta [Regioni ed endpoint](#) in Riferimenti generali AWS .

Per tutti Regioni AWS, gli AWS SDK utilizzano Signature Version 4 per impostazione predefinita per autenticare le richieste. Quando si utilizzano AWS SDK rilasciati prima di maggio 2016, potrebbe essere necessario richiedere la versione 4 di Signature, come mostrato nella tabella seguente.

SDK	Richiesta di Signature Version 4 per autenticazione delle richieste
AWS CLI	Per il profilo di default, eseguire il comando indicato di seguito: <pre>\$ aws configure set default.s3.signature_version s3v4</pre> Per un profilo personalizzato, eseguire il comando indicato di seguito: <pre>\$ aws configure set profile.your_profile_name.s3.signature_version s3v4</pre>
SDK Java	Aggiungere quanto segue al codice: <pre>System.setProperty(SDKGlobalConfiguration.ENABLE_S3_SIGV4_SYSTEM_PROPERTY, "true");</pre> In alternativa, specificare quanto segue nella riga di comando: <pre>-Dcom.amazonaws.services.s3.enableV4</pre>
SDK PHP	Impostare il parametro <code>signature</code> su <code>v4</code> durante la creazione del client del servizio Amazon S3 per l'SDK PHP v2: <pre><?php</pre>

SDK	Richiesta di Signature Version 4 per autenticazione delle richieste <pre>\$client = S3Client::factory(['region' => 'YOUR-REGION', 'version' => 'latest', 'signature' => 'v4']);</pre> Quando si usa l'SDK PHP v3, impostare il parametro <code>signature_version</code> su <code>v4</code> durante la creazione del client del servizio Amazon S3: <pre><?php \$s3 = new Aws\S3\S3Client(['version' => '2006-03-01', 'region' => 'YOUR-REGION', 'signature_version' => 'v4']);</pre>
Python-Boto SDK	Specificare quanto segue nel file di configurazione boto di default: <pre>[s3] use-sigv4 = True</pre>
SDK Ruby	SDK Ruby - Versione 1: impostare il parametro <code>:s3_signature_version</code> su <code>:v4</code> durante la creazione del client: <pre>s3 = AWS::S3::Client.new(:s3_signature_version => :v4)</pre> SDK Ruby - Versione 3: impostare il parametro <code>signature_version</code> su <code>v4</code> durante la creazione del client: <pre>s3 = Aws::S3::Client.new(signature_version: 'v4')</pre>

SDK	Richiesta di Signature Version 4 per autenticazione delle richieste
SDK .NET	Aggiungere quanto segue al codice prima di creare il client Amazon S3: <pre>AWSConfigsS3.UseSignatureVersion4 = true;</pre> In alternativa, aggiungere quanto segue al file di configurazione: <pre><appSettings> <add key="AWS.S3.UseSignatureVersion4" value="true" /> </appSettings></pre>

AWS Signature Version 2 disattivata (obsoleta) per Amazon S3

Signature Version 2 sta per essere disattivato perché obsoleto in Amazon S3. In seguito, Amazon S3 accetterà solo richieste API firmate con Signature Version 4.

Questa sezione fornisce le risposte alle domande più frequenti sulla fine del supporto di Signature Version 2.

Cos'è la versione 2/4 Signature e cosa significa firmare le richieste?

Il processo di firma Signature Version 2 o Signature Version 4 viene utilizzato per autenticare le richieste API Amazon S3. La firma delle richieste consente ad Amazon S3 di identificare il mittente della richiesta e protegge le richieste dalle azioni di utenti malintenzionati.

Per ulteriori informazioni sulla firma AWS delle richieste, consulta la sezione [Signing AWS API Requests](#) in Riferimenti generali di AWS.

Quali aggiornamenti vengono introdotti?

Attualmente, sono supportate le richieste API Amazon S3 firmate con i processi Signature Version 2 e Signature Version 4. In seguito, Amazon S3 accetterà solo le richieste firmate con Signature Version 4.

Per ulteriori informazioni sulla firma AWS delle richieste, vedere [Modifiche nella versione 4 di Signature](#) in Riferimenti generali di AWS.

Perché è stato rilasciato questo aggiornamento?

Signature Version 4 fornisce una maggiore sicurezza perché utilizza una chiave di firma invece della chiave di accesso segreta. La versione 4 di Signature è attualmente supportata in tutte le aree Regioni AWS, mentre la versione 2 di Signature è supportata solo nelle regioni lanciate prima di gennaio 2014. Questo aggiornamento consente di offrire un'esperienza più uniforme in tutte le regioni.

Come posso verificare di stare utilizzando Signature Version 4 e di quali aggiornamenti ho bisogno?

La versione della firma utilizzata per firmare le richieste viene generalmente impostata dallo strumento o dall'SDK sul lato client. Per impostazione predefinita, le ultime versioni dei nostri AWS SDK utilizzano la versione 4 di Signature. Per i software di terza parte, contatta il team di supporto specifico per il tuo software per verificare la versione richiesta. Se stai inviando chiamate REST ad Amazon S3, devi modificare l'applicazione per utilizzare il processo di firma Signature Version 4.

Per informazioni sulla versione degli AWS SDK da utilizzare per passare alla versione 4 di Signature, consulta [Passaggio da Signature Version 2 a Signature Version 4](#)

Per ulteriori informazioni sull'uso di Signature Version 4 con l'API REST di Amazon S3, consulta [Autenticazione delle richieste \(AWS Signature Version 4\)](#) nella Documentazione di riferimento delle API di Amazon Simple Storage Service.

Cosa succede se non eseguo gli aggiornamenti?

Le richieste firmate con Signature Version 2 effettuate successivamente non verranno autenticate con Amazon S3. I richiedenti visualizzeranno degli errori che indicano che la richiesta deve essere firmata con Signature Version 4.

Devo apportare queste modifiche anche se sto utilizzando un URL prefirmato che richiede la firma per un periodo superiore a 7 giorni?

Se utilizzi un URL prefirmato che richiede la firma per un periodo superiore a 7 giorni, non è necessaria alcuna operazione. Puoi continuare a utilizzare AWS Signature Version 2 per firmare e autenticare l'URL prefirmato. Per maggiori dettagli su come migrare alla versione 4 di Signature per uno scenario di URL prefirmato, consulta la documentazione di Amazon S3.

Ulteriori informazioni

- Per ulteriori informazioni sull'utilizzo di Signature Version 4, consulta [Signing AWS API Requests](#).
- Visualizza l'elenco delle differenze tra Signature Version 2 e Signature Version 4 in [Modifiche di Signature Version 4](#).
- Visualizza il post [AWS Signature Version 4 per sostituire AWS Signature Version 2 per la firma delle richieste API Amazon S3](#) nei AWS forum.
- Per domande o dubbi, contatta [Supporto](#).

Passaggio da Signature Version 2 a Signature Version 4

Se al momento utilizzi Signature Version 2 per l'autenticazione delle richieste API Amazon S3, devi passare a Signature Version 4. Il supporto per Signature Version 2 sta per terminare, come descritto in [AWS Signature Version 2 disattivata \(obsoleta\) per Amazon S3](#).

Per ulteriori informazioni sull'uso di Signature Version 4 con l'API REST di Amazon S3, consulta [Autenticazione delle richieste \(AWS Signature Version 4\)](#) nella Documentazione di riferimento delle API di Amazon Simple Storage Service.

La tabella seguente elenca gli SDK con la versione minima necessaria per utilizzare Signature Version 4 (SigV4). Se si utilizzano URL prefirmati con gli SDK AWS Java, JavaScript (Node.js) o Python (Boto/CLI), è necessario impostare la versione 4 di Signature corretta Regione AWS nella configurazione del client. Per informazioni sull'impostazione di SigV4 nella configurazione del client, consulta [Specificare la versione della firma nell'autenticazione della richiesta](#).

Se usi questo SDK/Product	Esegui l'aggiornamento a questa versione dell'SDK	La modifica del codice nel client è necessaria per utilizzare Sigv4?	Collegamento alla documentazione sugli SDK
AWS SDK per Java v1	Aggiornamento a Java	Sì	Specificare la versione della firma nell'autenticazione della richiesta

Se usi questo SDK/Product	Esegui l'aggiornamento a questa versione dell'SDK	La modifica del codice nel client è necessaria per utilizzare Sigv4?	Collegamento alla documentazione sugli SDK
	1.11.201+ o v2.		
AWS SDK per Java v2	Non è necessari o l'aggiornamento dell'SDK.	No	AWS SDK per Java
AWS SDK per .NET v1	Aggiornamento alla versione 3.1.10 o successiva.	Sì	AWS SDK per .NET
AWS SDK per .NET v2	Aggiornamento alla versione 3.1.10 o successiva.	No	AWS SDK per .NET v2
AWS SDK per .NET v3	Aggiornamento alla versione 3.3.0.0 o successiva.	Sì	AWS SDK per .NET v3

Se usi questo SDK/Product	Esegui l'aggiornamento a questa versione dell'SDK	La modifica del codice nel client è necessaria per utilizzare Sigv4?	Collegamento alla documentazione sugli SDK
AWS SDK per JavaScript v1	Aggiornamento alla versione 2.68.0 o successiva.	Sì	AWS SDK per JavaScript
AWS SDK per JavaScript v2	Aggiornamento alla versione 2.68.0 o successiva.	Sì	AWS SDK per JavaScript
AWS SDK per JavaScript v3	Al momento, non è richiesta alcuna operazione. Aggiornamento alla versione principale V3 nel T3 2019.	No	AWS SDK per JavaScript

Se usi questo SDK/Product	Esegui l'aggiornamento a questa versione dell'SDK	La modifica del codice nel client è necessaria per utilizzare Sigv4?	Collegamento alla documentazione sugli SDK
AWS SDK per PHP v1	Consiglia to l'aggiornamento alla versione più recente di PHP o almeno alla v2.7.4 con il parametro di firma impostato su v4 nella configurazione del client S3.	Sì	AWS SDK per PHP

Se usi questo SDK/Product	Esegui l'aggiornamento a questa versione dell'SDK	La modifica del codice nel client è necessaria per utilizzare Sigv4?	Collegamento alla documentazione sugli SDK
AWS SDK per PHP v2	Consiglia to l'aggiornamento alla versione più recente di PHP o almeno alla v2.7.4 con il parametro di firma impostato su v4 nella configurazione del client S3.	No	AWS SDK per PHP
AWS SDK per PHP v3	Non è necessari o l'aggiornamento dell'SDK.	No	AWS SDK per PHP
Boto2	Aggiornam ento a Boto2 v2.49.0.	Sì	Aggiornamento a Boto 2

Se usi questo SDK/Product	Esegui l'aggiornamento a questa versione dell'SDK	La modifica del codice nel client è necessaria per utilizzare Sigv4?	Collegamento alla documentazione sugli SDK
Boto3	Aggiornamento alla versione 1.5.71 (Botocore), 1.4.6 (Boto3).	Sì	Boto 3 - AWS SDK per Python
AWS CLI	Aggiornamento alla versione 1.11.108.	Sì	AWS Command Line Interface
AWS CLI v2 (anteprima)	Non è necessario o l'aggiornamento dell'SDK.	No	AWS Command Line Interface versione 2
AWS SDK per Ruby v1	Aggiornamento a Ruby V3.	Sì	Ruby V3 per AWS
AWS SDK per Ruby v2	Aggiornamento a Ruby V3.	Sì	Ruby V3 per AWS
AWS SDK per Ruby v3	Non è necessario o l'aggiornamento dell'SDK.	No	Ruby V3 per AWS

Se usi questo SDK/Product	Esegui l'aggiornamento a questa versione dell'SDK	La modifica del codice nel client è necessaria per utilizzare Sigv4?	Collegamento alla documentazione sugli SDK
Go	Non è necessari o l'aggiornamento dell'SDK.	No	AWS SDK per Go
C++	Non è necessari o l'aggiornamento dell'SDK.	No	AWS SDK per C++

AWS Tools for Windows PowerShell or AWS Tools for PowerShell Core

Se utilizzi versioni del modulo precedenti alla 3.3.0.0, devi eseguire l'aggiornamento alla versione 3.3.0.0.

Per ottenere informazioni sulla versione, utilizza il cmdlet `Get-Module`:

```
Get-Module -Name AWSPowershell
Get-Module -Name AWSPowershell.NetCore
```

Per eseguire l'aggiornamento alla versione 3.3.0.0, utilizza il cmdlet `Update-Module`:

```
Update-Module -Name AWSPowershell
Update-Module -Name AWSPowershell.NetCore
```

Puoi utilizzare gli URL prefirmati, validi per più di 7 giorni, su cui invierai il traffico di Signature Version 2.

Ottenere gli ID di richiesta Amazon S3 per Supporto AWS

Ogni volta che contatti Supporto AWS perché hai riscontrato errori o comportamenti imprevisti in Amazon S3, devi fornire gli ID di richiesta associati all'azione non riuscita. Supporto AWS utilizza questi ID di richiesta per aiutarti a risolvere i problemi che stai riscontrando.

Gli ID richiesta vengono forniti in coppia, vengono restituiti in ogni risposta elaborata da Amazon S3 (anche quelle vere) ed è possibile accedervi tramite i log dettagliati. Esistono diversi metodi comuni per ottenere gli ID delle richieste, inclusi i log di accesso al server S3 e AWS CloudTrail gli eventi o gli eventi relativi ai dati.

Dopo aver recuperato questi registri, copia e conserva questi due valori, perché ti serviranno al momento del contatto Supporto AWS. Per informazioni su come contattare Supporto AWS, consulta [Contatti AWS](#) o [Supporto AWS Documentazione](#).

Argomenti

- [Utilizzo di AWS CLI per ottenere gli ID delle richieste](#)
- [Utilizzo di Windows PowerShell per ottenere gli ID delle richieste](#)
- [Utilizzo AWS CloudTrail eventi relativi ai dati per ottenere gli ID delle richieste](#)
- [Utilizzo della registrazione degli accessi al server S3 per recuperare gli ID richiesta](#)
- [Utilizzo di HTTP per recuperare gli ID richiesta](#)
- [Utilizzo di un browser Web per recuperare gli ID richiesta](#)
- [Utilizzo di AWS SDK per ottenere gli ID delle richieste](#)

Utilizzo di AWS CLI per ottenere gli ID delle richieste

Per ottenere gli ID delle richieste quando si utilizza AWS Command Line Interface (AWS CLI), --debug aggiungili al comando. Ad esempio, dovresti vedere x-amz-request-id and x-amz-id-2 nel registro di debug come mostrato di seguito:

```
...
2025-04-30 14:35:28,572 - MainThread - botocore.parsers - DEBUG - Response headers:
{'x-amz-id-2': 'a+zm50vDJH3LLmCiXvwEo0u0PtPS/qCJaBvB2ZMH9dzyzTiJhiLZkBFfOfsPf0KztUKT/
garCI=', 'x-amz-request-id': 'N4NMN0MJ4VDFZMX9', 'Date': 'Wed, 30 Apr 2025 21:35:29
```

```
GMT', 'Content-Type': 'application/xml', 'Transfer-Encoding': 'chunked', 'Server':  
'AmazonS3'}  
...
```

Utilizzo di Windows PowerShell per ottenere gli ID delle richieste

Per informazioni sul recupero dei log con Windows PowerShell, consulta la sezione [Response Logging AWS Tools for Windows PowerShell](#) in the AWS Developer Tools Blog.

Utilizzo AWS CloudTrail eventi relativi ai dati per ottenere gli ID delle richieste

Un bucket Amazon S3 configurato con eventi di CloudTrail dati per registrare le operazioni API a livello di oggetto di S3 fornisce informazioni dettagliate sulle azioni intraprese da un utente, ruolo o servizio in Amazon S3. AWS Puoi identificare gli ID delle richieste S3 interrogando gli eventi con Athena. CloudTrail

Per ulteriori informazioni, consulta [Identificazione delle richieste Amazon S3 utilizzando CloudTrail](#) nella Amazon Simple Storage Service User Guide.

Utilizzo della registrazione degli accessi al server S3 per recuperare gli ID richiesta

Un bucket Amazon S3 configurato per la registrazione degli accessi al server S3 fornisce record dettagliati per le richieste effettuate a tale bucket. È possibile identificare gli ID delle richieste S3 eseguendo query sui log degli accessi al server utilizzando Athena.

Per ulteriori informazioni, consulta [Interrogare i log di accesso per le richieste utilizzando Amazon Athena](#) nella Amazon Simple Storage Service User Guide.

Utilizzo di HTTP per recuperare gli ID richiesta

È possibile recuperare gli ID richiesta, `x-amz-request-id` e `x-amz-id-2` registrando le parti di una richiesta HTTP prima che raggiunga l'applicazione di destinazione. Esistono diversi strumenti di terze parti che è possibile utilizzare per recuperare i log verbose per le richieste HTTP. Scegli lo strumento che preferisci usare ed eseguilo per rimanere in ascolto sulla porta che gestisce il traffico di Amazon S3 mentre invii un'altra richiesta HTTP Amazon S3.

Per le richieste HTTP, la coppia di ID richiesta sarà simile agli esempi seguenti:

```
x-amz-request-id: 79104EXAMPLEB723
x-amz-id-2: IOWQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km
```

Note

Le richieste HTTPS sono crittografate e nascoste nella maggior parte delle acquisizioni dei pacchetti.

Utilizzo di un browser Web per recuperare gli ID richiesta

La maggiore dei browser Web include strumenti per sviluppatori che puoi usare per visualizzare le intestazioni delle richieste.

Per le richieste basate sul Web che restituiscono un errore, la coppia di ID richiesta sarà simile agli esempi seguenti.

```
<Error><Code>AccessDenied</Code><Message>Access Denied</Message>
<RequestId>79104EXAMPLEB723</RequestId><HostId>IOWQ4fDEXAMPLEQM
+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km</HostId></Error>
```

Per ottenere la coppia di ID richiesta proveniente da richieste completate, utilizza gli strumenti per sviluppatori del browser per esaminare le intestazioni delle risposte HTTP.

Utilizzo di AWS SDK per ottenere gli ID delle richieste

Le sezioni seguenti includono informazioni per configurare la registrazione utilizzando diversi SDK. AWS

Note

Sebbene sia possibile abilitare la registrazione dettagliata su ogni richiesta e risposta, non è consigliabile abilitare l'accesso ai sistemi di produzione, poiché richieste o risposte di grandi dimensioni possono rallentare notevolmente un'applicazione.

Per le richieste AWS SDK, la coppia di ID di richiesta sarà simile alla seguente:

```
Status Code: 403, AWS Service: Amazon S3, AWS Request ID: 79104EXAMPLEB723
```



```
AWS Error Code: AccessDenied  AWS Error Message: Access Denied
S3 Extended Request ID: IOWQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK
+Jd1vEXAMPLEa3Km
```

C++

Il tipo di logger e la verbosità sono specificati durante l'inizializzazione dell'SDK nell'argomento. `SDKOptions` L'esempio seguente specifica il livello di verbosità come. `LogLevel::Debug`

Il logger predefinito scriverà nel filesystem e il file viene denominato utilizzando la seguente convenzione. `aws_sdk_YYYY-MM-DD-HH.log` Il logger crea un nuovo file ogni ora.

```
Aws::SDKOptions options;
options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
Aws::InitAPI(options);
// ...
Aws::ShutdownAPI(options);
```

Per ulteriori informazioni, vedi [Come si attiva la registrazione?](#) nel wiki AWS SDK per C++ attivo. GitHub

Go

È possibile configurare la registrazione utilizzando SDK for Go. Per ulteriori informazioni, consulta Logging [in the](#) AWS SDK per Go v2 Developer Guide.

Java

È possibile abilitare la registrazione di richieste o risposte specifiche per rilevare e restituire solo le intestazioni pertinenti. A tale scopo, importare la classe `com.amazonaws.services.s3.S3ResponseMetadata`. È quindi possibile archiviare la richiesta in una variabile prima di eseguire la richiesta effettiva. Per recuperare la richiesta o la risposta registrata, chiama `getCachedResponseMetadata(AmazonWebServiceRequest request).getRequestID()`.

```
PutObjectRequest req = new PutObjectRequest(bucketName, key, createSampleFile());
s3.putObject(req);
S3ResponseMetadata md = s3.getCachedResponseMetadata(req);
System.out.println("Host ID: " + md.getHostId() + " RequestID: " +
    md.getRequestId());
```

In alternativa, è possibile utilizzare la registrazione verbose di ogni richiesta e risposta Java. Per ulteriori informazioni, consulta l'argomento relativo alla [registrazione dettagliata in rete](#) nella Guida per gli sviluppatori di AWS SDK per Java .

JavaScript

AWS SDK per JavaScript Dispone di un logger integrato in modo da poter registrare le chiamate API effettuate con esso. Per attivare il logger e stampare le voci di registro nella console, aggiungi la seguente dichiarazione al tuo codice:

```
AWS.config.logger = console;
```

Per ulteriori informazioni, consulta [Registrazione delle AWS SDK per JavaScript chiamate](#) nella Guida per gli AWS SDK per JavaScript sviluppatori.

Kotlin

Con AWS SDK per Kotlin, puoi specificare la modalità di registro per i messaggi a livello cablato utilizzando il codice o le impostazioni dell'ambiente. È possibile impostare la modalità di registro per le richieste HTTP e le risposte HTTP.

Per attivare la registrazione aggiuntiva, imposta la `logMode` proprietà quando crei un client di servizio:

```
import aws.smithy.kotlin.runtime.client.LogMode

// ...

val client = S3Client {
    // ...
    logMode = LogMode.LogRequestWithBody + LogMode.LogResponse
}
```

In alternativa, puoi impostare la modalità di registro utilizzando una variabile di ambiente:

```
export SDK_LOG_MODE=LogRequestWithBody|LogResponse
```

Per ulteriori informazioni, consulta [Logging](#) in the AWS SDK per Kotlin Developer Guide.

.NET

È possibile configurare la registrazione con SDK per .NET utilizzando lo strumento di System.Diagnostics registrazione integrato. Per ulteriori informazioni, consulta il post [Logging with the SDK per .NET](#) AWS Developer Blog.

Note

Per default, il log restituito contiene solo informazioni sugli errori. Per recuperare gli ID richiesta, è necessario aggiungere `AWSLogMetrics` (e facoltativamente `AWSResponseLogging`) al file di configurazione.

PHP

È possibile ottenere informazioni di debug, inclusi i dati inviati via cavo, impostando l'opzione debug su `true` in un costruttore client.

```
$s3Client = new Aws\S3\S3Client([
    'region' => 'us-standard',
    'version' => '2006-03-01',
    'debug'   => true
]);
```

Per ulteriori informazioni, consulta [Come è possibile controllare i dati che vengono trasmessi in rete?](#) nella Guida per gli sviluppatori di AWS SDK per PHP .

Python (Boto3)

Con il AWS SDK per Python (Boto3), puoi registrare risposte specifiche. È possibile utilizzare questa funzione per acquisire solo le intestazioni pertinenti. Il codice seguente mostra come registrare parti della risposta a un file:

```
import logging
import boto3
logging.basicConfig(filename='logfile.txt', level=logging.INFO)
logger = logging.getLogger(__name__)
s3 = boto3.resource('s3')
response = s3.Bucket(bucket_name).Object(object_key).put()
logger.info("HTTPStatusCode: %s", response['ResponseMetadata']['HTTPStatusCode'])
```

```
logger.info("RequestId: %s", response['ResponseMetadata']['RequestId'])
logger.info("HostId: %s", response['ResponseMetadata']['HostId'])
logger.info("Date: %s", response['ResponseMetadata']['HTTPHeaders']['date'])
```

È inoltre possibile rilevare le eccezioni e registrare le informazioni pertinenti quando viene sollevata un'eccezione. Per ulteriori informazioni, consulta l'argomento relativo alla [ricerca di informazioni utili nelle risposte degli errori](#) nella pagina Riferimento API per l'SDK AWS per Python (Boto).

Inoltre, è possibile configurare Boto3 per l'output dei log di debug dettagliati utilizzando il seguente codice:

```
import logging
import boto3
boto3.set_stream_logger('', logging.DEBUG)
```

Per ulteriori informazioni, consulta [set_stream_logger](#) nella pagina Riferimento API per l'SDK AWS per Python (Boto).

Ruby

Puoi ottenere gli ID delle tue richieste utilizzando l'SDK per Ruby versione 3. Puoi abilitare la registrazione via cavo HTTP nel tuo client utilizzando il seguente codice:

```
s3 = Aws::S3::Client.new(http_wire_trace: true)
```

Note

La registrazione telefonica può includere informazioni sensibili, come l'ID della chiave di accesso. Le informazioni sensibili devono essere disinfettate prima di condividerle con AWS l'assistenza.

Puoi anche trovare l'ID della richiesta dell'oggetto del contesto della richiesta nella risposta alla richiesta o all'errore:

```
# Finding the request ID from an error:
begin
  s3.put_object(bucket: 'bucket', key: 'key', body: 'test')
```

```
rescue Aws::S3::Errors::ServiceError => e
  puts e.context[:request_id]
  puts e.context[:s3_host_id]
end

# Finding the request ID from a successful call:
resp = s3.put_object(bucket: 'bucket', key: 'key', body: 'test')
puts resp.context[:request_id]
puts resp.context[:s3_host_id]
```

Per ulteriori informazioni, consulta [Debugging using wire trace information from an AWS SDK per Ruby client](#) nella AWS SDK per Ruby Developer Guide.

Rust

Per abilitare la registrazione, aggiungete il `tracing-subscriber` crate e inizializzatelo nella vostra applicazione Rust.

Aggiungi la libreria di tracciamento al tuo file: `Cargo.toml`

```
tracing-subscriber = { version = "0.3", features = ["env-filter"] }
```

Quindi, nel tuo codice Rust, inizializza il logger nella funzione principale prima di chiamare qualsiasi operazione SDK:

```
tracing_subscriber::fmt::init();
```

Per ulteriori informazioni, consulta [Configurazione e utilizzo della registrazione nella Guida per gli AWS SDK per Rust](#) sviluppatori. AWS SDK per Rust

Operazioni API a livello di oggetto di Amazon S3 supportate per tabelle S3

La tabella seguente include le operazioni API S3 supportate a livello di oggetto e le intestazioni corrispondenti per le tabelle S3. Per un elenco delle API di Amazon S3 Tables, consulta Amazon S3 Tables. https://docs.aws.amazon.com/AmazonS3/latest/API/API_Operations_Amazon_S3_Tables.html Per ulteriori informazioni su Amazon S3 Tables, consulta [Working with Amazon S3 Tables and table buckets](#) nella Amazon S3 User Guide.

Object-level API	Intestazioni supportate	Note	
AbortMultipartUpload	x-amz-expected-bucket-owner	Nessuno	
CompleteMultipartUpload	x-amz-checksum-crc32	Nessuna	
	x-amz-checksum-crc32c	Nessuna	
	x-amz-checksum-sha1	Nessuna	
	x-amz-checksum-sha256	Nessuna	
	x-amz-expected-bucket-owner	Nessuna	
	If-Match	Nessuna	
	If-None-Match	Nessuno	
CreateMultipartUpload	x-amz-acl: ACL	Per le tabelle S3, il valore predefinito è <code>bucket-owner-full-control</code> e non può essere modificato.	
	Cache-Control	Nessuno	
	Content-Disposition	Nessuna	
	Content-Encoding	Nessuna	

Object-level API	Intestazioni supportate	Note	
	Content-Language	Nessuna	
	Content-Type	Nessuna	
	Expires	Nessuna	
	x-amz-checksum-algorithm	Nessuno	
	x-amz-storage-class	Per S3 Tables, il valore predefinito è STANDARD e non può essere modificato.	
	x-amz-server-side-encryption	Per S3 Tables, il valore predefinito è (SSE-S3) (AES256) e non può essere modificato.	
GetObject	If-Match	Nessuno	
	If-Modified-Since	Nessuna	
	If-None-Match	Nessuna	
	If-Unmodified-Since	Nessuna	
	Range	Nessuna	
	x-amz-expected-bucket-owner	Nessuna	
	x-amz-checksum-mode	Nessuna	

Object-level API	Intestazioni supportate	Note	
HeadObject	If-Match	Nessuna	
	If-Modified-Since	Nessuna	
	If-None-Match	Nessuna	
	If-Unmodified-Since	Nessuna	
	Range	Nessuna	
	x-amz-checksum-mode	Nessuna	
	x-amz-expected-bucket-owner	Nessuna	
ListParts	x-amz-expected-bucket-owner	Nessuno	
PutObject	x-amz-acl: ACL	Per S3 Tables, il valore predefinito è bucket-owner-full-control e non può essere modificato.	
	Cache-Control	Nessuno	
	Content-Disposition	Nessuna	
	Content-Encoding	Nessuna	
	Content-Language	Nessuna	
	Content-Length	Nessuna	

Object-level API	Intestazioni supportate	Note	
	Content-MD5	Nessuna	
	Content-Type	Nessuna	
	x-amz-sdk-checksum-algorithm	Nessuna	
	x-amz-checksum-crc32	Nessuna	
	x-amz-checksum-crc32c	Nessuna	
	x-amz-checksum-sha1	Nessuna	
	x-amz-checksum-sha256	Nessuna	
	Expires	Nessuna	
	If-None-Match	Nessuna	
	x-amz-expected-bucket-owner	Nessuno	
	x-amz-storage-class	Per S3 Tables, il valore predefinito è STANDARD e non può essere modificato.	
	x-amz-server-side-encryption	Per S3 Tables, il valore predefinito è (SSE-S3) (AES256) e non può essere modificato.	

Object-level API	Intestazioni supportate	Note	
UploadPart	Content-Length	Nessuno	
	Content-MD5	Nessuna	
	x-amz-checksum-crc32	Nessuna	
	x-amz-checksum-crc32c	Nessuna	
	x-amz-checksum-sha1	Nessuna	
	x-amz-checksum-sha256	Nessuna	
	x-amz-expected-bucket-owner	Nessuno	

Esempi di codice per l'utilizzo di Amazon S3 AWS SDK

I seguenti esempi di codice mostrano come usare Amazon S3 con un kit di sviluppo AWS software (SDK).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Esempi di codice

- [Esempi di codice per l'utilizzo di Amazon S3 AWS SDK](#)
- [Esempi di base per l'utilizzo di Amazon S3 AWS SDK](#)
 - [Hello Amazon S3](#)
 - [Scopri le nozioni di base di Amazon S3 con un AWS SDK](#)
 - [Azioni per l'utilizzo di Amazon S3 AWS SDK](#)
 - [Da utilizzare AbortMultipartUploadcon un AWS SDK o CLI](#)
 - [Utilizzare CompleteMultipartUploadcon un AWS SDK o CLI](#)
 - [Utilizzare CopyObjectcon un AWS SDK o CLI](#)
 - [Utilizzare CreateBucketcon un AWS SDK o CLI](#)
 - [Utilizzare CreateMultiRegionAccessPointcon un AWS SDK](#)
 - [Utilizzare CreateMultipartUploadcon un AWS SDK o CLI](#)
 - [Utilizzare CreatePresignedPostcon un AWS SDK](#)
 - [Utilizzare DeleteBucketcon un AWS SDK o CLI](#)
 - [Utilizzare DeleteBucketAnalyticsConfiguration con una CLI](#)
 - [Utilizzare DeleteBucketCorscon un AWS SDK o CLI](#)
 - [Utilizzare DeleteBucketEncryption con una CLI](#)
 - [Utilizzare DeleteBucketInventoryConfiguration con una CLI](#)
 - [Utilizzare DeleteBucketLifecyclecon un AWS SDK o CLI](#)
 - [Utilizzare DeleteBucketMetricsConfiguration con una CLI](#)
 - [Utilizzare DeleteBucketPolicycon un AWS SDK o CLI](#)
 - [Utilizzare DeleteBucketReplication con una CLI](#)
 - [Utilizzare DeleteBucketTagging con una CLI](#)

- [Utilizzare DeleteBucketWebsite con un AWS SDK o CLI](#)
- [Utilizzare DeleteObject con un AWS SDK o CLI](#)
- [Utilizzare DeleteObjectTagging con una CLI](#)
- [Utilizzare DeleteObjects con un AWS SDK o CLI](#)
- [Utilizzare DeletePublicAccessBlock con una CLI](#)
- [Utilizzare GetBucketAccelerateConfiguration con una CLI](#)
- [Utilizzare GetBucketAcl con un AWS SDK o CLI](#)
- [Utilizzare GetBucketAnalyticsConfiguration con una CLI](#)
- [Utilizzare GetBucketCors con un AWS SDK o CLI](#)
- [Utilizzare GetBucketEncryption con un AWS SDK o CLI](#)
- [Utilizzare GetBucketInventoryConfiguration con una CLI](#)
- [Utilizzare GetBucketLifecycleConfiguration con un AWS SDK o CLI](#)
- [Utilizzare GetBucketLocation con un AWS SDK o CLI](#)
- [Utilizzare GetBucketLogging con una CLI](#)
- [Utilizzare GetBucketMetricsConfiguration con una CLI](#)
- [Utilizzare GetBucketNotification con una CLI](#)
- [Utilizzare GetBucketPolicy con un AWS SDK o CLI](#)
- [Utilizzare GetBucketPolicyStatus con una CLI](#)
- [Utilizzare GetBucketReplication con un AWS SDK o CLI](#)
- [Utilizzare GetBucketRequestPayment con una CLI](#)
- [Utilizzare GetBucketTagging con una CLI](#)
- [Utilizzare GetBucketVersioning con una CLI](#)
- [Utilizzare GetBucketWebsite con un AWS SDK o CLI](#)
- [Utilizzare GetObject con un AWS SDK o CLI](#)
- [Utilizzare GetObjectAcl con un AWS SDK o CLI](#)
- [Utilizzare GetObjectAttributes con un AWS SDK o CLI](#)
- [Utilizzare GetObjectLegalHold con un AWS SDK o CLI](#)
- [Utilizzare GetObjectLockConfiguration con un AWS SDK o CLI](#)
- [Utilizzare GetObjectRetention con un AWS SDK o CLI](#)
- [Utilizzare GetObjectTagging con una CLI](#)

- [Utilizzare GetPublicAccessBlock con una CLI](#)
- [Utilizzare HeadBucketcon un AWS SDK o CLI](#)
- [Utilizzare HeadObjectcon un AWS SDK o CLI](#)
- [Utilizzare ListBucketAnalyticsConfigurations con una CLI](#)
- [Utilizzare ListBucketInventoryConfigurations con una CLI](#)
- [Utilizzare ListBucketscon un AWS SDK o CLI](#)
- [Utilizzare ListMultipartUploadscon un AWS SDK o CLI](#)
- [Utilizzare ListObjectVersionscon un AWS SDK o CLI](#)
- [Utilizzare ListObjects con una CLI](#)
- [Usa ListObjectsV2 con un AWS SDK o CLI](#)
- [Utilizzare PutBucketAccelerateConfigurationcon un AWS SDK o CLI](#)
- [Utilizzare PutBucketAclcon un AWS SDK o CLI](#)
- [Utilizzare PutBucketCorscon un AWS SDK o CLI](#)
- [Utilizzare PutBucketEncryptioncon un AWS SDK o CLI](#)
- [Utilizzare PutBucketLifecycleConfigurationcon un AWS SDK o CLI](#)
- [Utilizzare PutBucketLoggingcon un AWS SDK o CLI](#)
- [Utilizzare PutBucketNotification con una CLI](#)
- [Utilizzare PutBucketNotificationConfigurationcon un AWS SDK o CLI](#)
- [Utilizzare PutBucketPolicycon un AWS SDK o CLI](#)
- [Utilizzare PutBucketReplicationcon un AWS SDK o CLI](#)
- [Utilizzare PutBucketRequestPayment con una CLI](#)
- [Utilizzare PutBucketTagging con una CLI](#)
- [Utilizzare PutBucketVersioningcon un AWS SDK o CLI](#)
- [Utilizzare PutBucketWebsitecon un AWS SDK o CLI](#)
- [Utilizzare PutObjectcon un AWS SDK o CLI](#)
- [Utilizzare PutObjectAclcon un AWS SDK o CLI](#)
- [Utilizzare PutObjectLegalHoldcon un AWS SDK o CLI](#)
- [Utilizzare PutObjectLockConfigurationcon un AWS SDK o CLI](#)
- [Utilizzare PutObjectRetentioncon un AWS SDK o CLI](#)
- [Utilizzare RestoreObjectcon un AWS SDK o CLI](#)

- [Utilizzare SelectObjectContent con un AWS SDK o CLI](#)
- [Utilizzare UploadPart con un AWS SDK o CLI](#)
- [Utilizzare UploadPartCopy con un AWS SDK o CLI](#)
- [Scenari per l'utilizzo di Amazon S3 AWS SDK](#)
 - [Controllare se un bucket esiste](#)
 - [Controlla se esiste un oggetto](#)
 - [Convertire il testo in voce e viceversa utilizzando un AWS SDK](#)
 - [Crea un URL predefinito per Amazon S3 utilizzando un AWS SDK](#)
 - [Creazione di un'applicazione di gestione delle risorse fotografiche che consente agli utenti di gestire le foto utilizzando etichette](#)
 - [Una pagina Web che elenca gli oggetti Amazon S3 utilizzando un AWS SDK](#)
 - [Creazione di un'applicazione Amazon Textract explorer](#)
 - [Elimina tutti gli oggetti in un determinato bucket Amazon S3 utilizzando un AWS SDK](#)
 - [Eliminare caricamenti multipart incompleti su Amazon S3 utilizzando un AWS SDK](#)
 - [Rileva i DPI nelle immagini con Amazon Rekognition utilizzando un AWS SDK](#)
 - [Rileva le entità nel testo estratto da un'immagine utilizzando un AWS SDK](#)
 - [Rileva i volti in un'immagine utilizzando un AWS SDK](#)
 - [Rileva oggetti nelle immagini con Amazon Rekognition utilizzando un AWS SDK](#)
 - [Rileva persone e oggetti in un video con Amazon Rekognition utilizzando un AWS SDK](#)
 - [Scaricare l'oggetto S3 "directory" da un bucket Amazon Simple Storage Service \(Amazon S3\)](#)
 - [Scaricare tutti gli oggetti da un bucket Amazon Simple Storage Service \(Amazon S3\) in una directory locale](#)
 - [Scarica uno stream di dimensioni sconosciute da un oggetto Amazon S3 utilizzando un AWS SDK](#)
 - [Ottieni un oggetto Amazon S3 da un punto di Multi-Region accesso utilizzando un AWS SDK](#)
 - [Ottieni un oggetto da un bucket Amazon S3 utilizzando un AWS SDK, specificando un'intestazione If-Modified-Since](#)
 - [Inizia a utilizzare la crittografia per oggetti Amazon S3 utilizzando un AWS SDK](#)
 - [Inizia a usare i tag per gli oggetti Amazon S3 utilizzando un AWS SDK](#)

- [Guida introduttiva ai cluster di elaborazione dei Big Data](#)
- [Guida introduttiva alla gestione della configurazione](#)
- [Guida introduttiva all'estrazione del testo dei documenti](#)
- [Guida introduttiva agli store di funzionalità di machine learning](#)
- [Guida introduttiva allo storage di oggetti](#)
- [Guida introduttiva all'analisi delle query](#)
- [Lavora con le funzionalità di blocco degli oggetti di Amazon S3 utilizzando un AWS SDK](#)
- [Effettua richieste condizionali per Amazon S3 utilizzando un AWS SDK](#)
- [Gestisci gli elenchi di controllo degli accessi \(ACL\) per i bucket Amazon S3 utilizzando un AWS SDK](#)
- [Gestisci messaggi Amazon SQS di grandi dimensioni utilizzando Amazon S3 con un AWS SDK](#)
- [Gestisci oggetti Amazon S3 con versioni in batch con una funzione Lambda utilizzando un AWS SDK](#)
- [Analizza gli URI di Amazon S3 utilizzando un AWS SDK](#)
- [Esegui una copia multiparte di un oggetto Amazon S3 utilizzando un AWS SDK](#)
- [Ricevi ed elabora le notifiche degli eventi di Amazon S3 utilizzando un AWS SDK](#)
- [Salvate EXIF e altre informazioni sull'immagine utilizzando un AWS SDK](#)
- [Invia notifiche di eventi S3 ad Amazon EventBridge utilizzando un AWS SDK](#)
- [Tieni traccia del caricamento o del download di un oggetto Amazon S3 utilizzando un AWS SDK](#)
- [Trasformare i dati per l'applicazione con S3 Object Lambda](#)
- [Esempi di approcci per test unitari e di integrazione con un AWS SDK](#)
- [Caricare in modo ricorsivo una directory locale in un bucket Amazon Simple Storage Service \(Amazon S3\)](#)
- [Carica o scarica file di grandi dimensioni da e verso Amazon S3 utilizzando un AWS SDK](#)
- [Carica un flusso di dimensioni sconosciute su un oggetto Amazon S3 utilizzando un AWS SDK](#)
- [Usa i checksum per lavorare con un oggetto Amazon S3 utilizzando un AWS SDK](#)
- [Lavora con le funzionalità di integrità degli oggetti di Amazon S3 utilizzando un AWS SDK](#)
- [Lavora con oggetti con versione di Amazon S3 utilizzando un AWS SDK](#)

- [Esempi serverless per Amazon S3](#)
 - [Invocazione di una funzione Lambda da un trigger Amazon S3](#)
- [Esempi di codice per l'utilizzo di Amazon S3 Control AWS SDK](#)
- [Esempi di base per l'utilizzo di Amazon S3 Control AWS SDK](#)
 - [Hello Amazon S3 Control](#)
 - [Scopri le nozioni di base di Amazon S3 Control con un AWS SDK](#)
 - [Azioni per Amazon S3 Control utilizzando AWS SDK](#)
 - [Da utilizzare CreateJobcon un AWS SDK o CLI](#)
 - [Utilizzare DeleteJobTaggingcon un AWS SDK](#)
 - [Utilizzare DescribeJobcon un AWS SDK o CLI](#)
 - [Utilizzare GetJobTaggingcon un AWS SDK](#)
 - [Utilizzare PutJobTaggingcon un AWS SDK](#)
 - [Utilizzare UpdateJobPrioritycon un AWS SDK o CLI](#)
 - [Utilizzare UpdateJobStatuscon un AWS SDK o CLI](#)
- [Esempi di codice per l'utilizzo di S3 Directory Bucket AWS SDK](#)
- [Esempi di base per l'utilizzo di S3 Directory Bucket AWS SDK](#)
 - [Hello Amazon S3 directory buckets](#)
 - [Impara le basi di S3 Directory Buckets con un AWS SDK](#)
 - [Azioni per l'utilizzo di S3 Directory Bucket AWS SDK](#)
 - [Utilizzare AbortMultipartUploadcon un AWS SDK](#)
 - [Utilizzare CompleteMultipartUploadcon un AWS SDK](#)
 - [Utilizzare CopyObjectcon un AWS SDK](#)
 - [Utilizzare CreateBucketcon un AWS SDK](#)
 - [Utilizzare CreateMultipartUploadcon un AWS SDK](#)
 - [Utilizzare CreateSessioncon un AWS SDK](#)
 - [Utilizzare DeleteBucketcon un AWS SDK](#)
 - [Utilizzare DeleteBucketEncryptioncon un AWS SDK](#)
 - [Utilizzare DeleteBucketPolicycon un AWS SDK](#)
 - [Utilizzare DeleteObjectcon un AWS SDK](#)
 - [Utilizzare DeleteObjectscon un AWS SDK](#)

- [Utilizzare GetBucketEncryption con un AWS SDK](#)
- [Utilizzare GetBucketPolicy con un AWS SDK](#)
- [Utilizzare GetObject con un AWS SDK](#)
- [Utilizzare GetObjectAttributes con un AWS SDK](#)
- [Utilizzare HeadBucket con un AWS SDK](#)
- [Utilizzare HeadObject con un AWS SDK](#)
- [Utilizzare ListDirectoryBuckets con un AWS SDK](#)
- [Utilizzare ListMultipartUploads con un AWS SDK](#)
- [Usa ListObjectsV2 con un AWS SDK](#)
- [Utilizzare ListParts con un AWS SDK](#)
- [Utilizzare PutBucketEncryption con un AWS SDK](#)
- [Utilizzare PutBucketPolicy con un AWS SDK](#)
- [Utilizzare PutObject con un AWS SDK](#)
- [Utilizzare UploadPart con un AWS SDK](#)
- [Utilizzare UploadPartCopy con un AWS SDK](#)
- [Scenari per l'utilizzo di S3 Directory Bucket AWS SDK](#)
 - [Crea un URL predefinito per i bucket di directory Amazon S3 per ottenere un oggetto utilizzando un AWS SDK](#)

Esempi di codice per l'utilizzo di Amazon S3 AWS SDK

I seguenti esempi di codice mostrano come usare Amazon S3 con un kit di sviluppo AWS software (SDK).

Nozioni di base: esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica chiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Per un elenco completo di guide ed esempi di codice per sviluppatori AWS SDK, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Simple Storage Service (Amazon S3)

- [Esempi di base per l'utilizzo di Amazon S3 AWS SDK](#)
 - [Hello Amazon S3](#)
 - [Scopri le nozioni di base di Amazon S3 con un AWS SDK](#)
 - [Azioni per l'utilizzo di Amazon S3 AWS SDK](#)
 - [Da utilizzare AbortMultipartUploadcon un AWS SDK o CLI](#)
 - [Utilizzare CompleteMultipartUploadcon un AWS SDK o CLI](#)
 - [Utilizzare CopyObjectcon un AWS SDK o CLI](#)
 - [Utilizzare CreateBucketcon un AWS SDK o CLI](#)
 - [Utilizzare CreateMultiRegionAccessPointcon un AWS SDK](#)
 - [Utilizzare CreateMultipartUploadcon un AWS SDK o CLI](#)
 - [Utilizzare CreatePresignedPostcon un AWS SDK](#)
 - [Utilizzare DeleteBucketcon un AWS SDK o CLI](#)
 - [Utilizzare DeleteBucketAnalyticsConfiguration con una CLI](#)
 - [Utilizzare DeleteBucketCorscon un AWS SDK o CLI](#)
 - [Utilizzare DeleteBucketEncryption con una CLI](#)
 - [Utilizzare DeleteBucketInventoryConfiguration con una CLI](#)
 - [Utilizzare DeleteBucketLifecyclecon un AWS SDK o CLI](#)
 - [Utilizzare DeleteBucketMetricsConfiguration con una CLI](#)
 - [Utilizzare DeleteBucketPolicycon un AWS SDK o CLI](#)
 - [Utilizzare DeleteBucketReplication con una CLI](#)
 - [Utilizzare DeleteBucketTagging con una CLI](#)
 - [Utilizzare DeleteBucketWebsitecon un AWS SDK o CLI](#)
 - [Utilizzare DeleteObjectcon un AWS SDK o CLI](#)
 - [Utilizzare DeleteObjectTagging con una CLI](#)
 - [Utilizzare DeleteObjectsccon un AWS SDK o CLI](#)
 - [Utilizzare DeletePublicAccessBlock con una CLI](#)

- [Utilizzare GetBucketAccelerateConfiguration con una CLI](#)
- [Utilizzare GetBucketAcl con un AWS SDK o CLI](#)
- [Utilizzare GetBucketAnalyticsConfiguration con una CLI](#)
- [Utilizzare GetBucketCors con un AWS SDK o CLI](#)
- [Utilizzare GetBucketEncryption con un AWS SDK o CLI](#)
- [Utilizzare GetBucketInventoryConfiguration con una CLI](#)
- [Utilizzare GetBucketLifecycleConfiguration con un AWS SDK o CLI](#)
- [Utilizzare GetBucketLocation con un AWS SDK o CLI](#)
- [Utilizzare GetBucketLogging con una CLI](#)
- [Utilizzare GetBucketMetricsConfiguration con una CLI](#)
- [Utilizzare GetBucketNotification con una CLI](#)
- [Utilizzare GetBucketPolicy con un AWS SDK o CLI](#)
- [Utilizzare GetBucketPolicyStatus con una CLI](#)
- [Utilizzare GetBucketReplication con un AWS SDK o CLI](#)
- [Utilizzare GetBucketRequestPayment con una CLI](#)
- [Utilizzare GetBucketTagging con una CLI](#)
- [Utilizzare GetBucketVersioning con una CLI](#)
- [Utilizzare GetBucketWebsite con un AWS SDK o CLI](#)
- [Utilizzare GetObject con un AWS SDK o CLI](#)
- [Utilizzare GetObjectAcl con un AWS SDK o CLI](#)
- [Utilizzare GetObjectAttributes con un AWS SDK o CLI](#)
- [Utilizzare GetObjectLegalHold con un AWS SDK o CLI](#)
- [Utilizzare GetObjectLockConfiguration con un AWS SDK o CLI](#)
- [Utilizzare GetObjectRetention con un AWS SDK o CLI](#)
- [Utilizzare GetObjectTagging con una CLI](#)
- [Utilizzare GetPublicAccessBlock con una CLI](#)
- [Utilizzare HeadBucket con un AWS SDK o CLI](#)
- [Utilizzare HeadObject con un AWS SDK o CLI](#)
- [Utilizzare ListBucketAnalyticsConfigurations con una CLI](#)
- [Utilizzare ListBucketInventoryConfigurations con una CLI](#)

- [Utilizzare ListBuckets con un AWS SDK o CLI](#)
- [Utilizzare ListMultipartUploads con un AWS SDK o CLI](#)
- [Utilizzare ListObjectVersions con un AWS SDK o CLI](#)
- [Utilizzare ListObjects con una CLI](#)
- [Usa ListObjectsV2 con un AWS SDK o CLI](#)
- [Utilizzare PutBucketAccelerateConfiguration con un AWS SDK o CLI](#)
- [Utilizzare PutBucketAcl con un AWS SDK o CLI](#)
- [Utilizzare PutBucketCors con un AWS SDK o CLI](#)
- [Utilizzare PutBucketEncryption con un AWS SDK o CLI](#)
- [Utilizzare PutBucketLifecycleConfiguration con un AWS SDK o CLI](#)
- [Utilizzare PutBucketLogging con un AWS SDK o CLI](#)
- [Utilizzare PutBucketNotification con una CLI](#)
- [Utilizzare PutBucketNotificationConfiguration con un AWS SDK o CLI](#)
- [Utilizzare PutBucketPolicy con un AWS SDK o CLI](#)
- [Utilizzare PutBucketReplication con un AWS SDK o CLI](#)
- [Utilizzare PutBucketRequestPayment con una CLI](#)
- [Utilizzare PutBucketTagging con una CLI](#)
- [Utilizzare PutBucketVersioning con un AWS SDK o CLI](#)
- [Utilizzare PutBucketWebsite con un AWS SDK o CLI](#)
- [Utilizzare PutObject con un AWS SDK o CLI](#)
- [Utilizzare PutObjectAcl con un AWS SDK o CLI](#)
- [Utilizzare PutObjectLegalHold con un AWS SDK o CLI](#)
- [Utilizzare PutObjectLockConfiguration con un AWS SDK o CLI](#)
- [Utilizzare PutObjectRetention con un AWS SDK o CLI](#)
- [Utilizzare RestoreObject con un AWS SDK o CLI](#)
- [Utilizzare SelectObjectContent con un AWS SDK o CLI](#)
- [Utilizzare UploadPart con un AWS SDK o CLI](#)
- [Utilizzare UploadPartCopy con un AWS SDK o CLI](#)

- [Scenari per l'utilizzo di Amazon S3 AWS SDK](#)

Simple Storage Service (Amazon S3)

- [Controllare se un bucket esiste](#)

- [Controlla se esiste un oggetto](#)
- [Convertire il testo in voce e viceversa utilizzando un AWS SDK](#)
- [Crea un URL predefinito per Amazon S3 utilizzando un AWS SDK](#)
- [Creazione di un'applicazione di gestione delle risorse fotografiche che consente agli utenti di gestire le foto utilizzando etichette](#)
- [Una pagina Web che elenca gli oggetti Amazon S3 utilizzando un AWS SDK](#)
- [Creazione di un'applicazione Amazon Textract explorer](#)
- [Elimina tutti gli oggetti in un determinato bucket Amazon S3 utilizzando un AWS SDK](#)
- [Eliminare caricamenti multiparte incompleti su Amazon S3 utilizzando un AWS SDK](#)
- [Rileva i DPI nelle immagini con Amazon Rekognition utilizzando un AWS SDK](#)
- [Rileva le entità nel testo estratto da un'immagine utilizzando un AWS SDK](#)
- [Rileva i volti in un'immagine utilizzando un AWS SDK](#)
- [Rileva oggetti nelle immagini con Amazon Rekognition utilizzando un AWS SDK](#)
- [Rileva persone e oggetti in un video con Amazon Rekognition utilizzando un AWS SDK](#)
- [Scaricare l'oggetto S3 "directory" da un bucket Amazon Simple Storage Service \(Amazon S3\)](#)
- [Scaricare tutti gli oggetti da un bucket Amazon Simple Storage Service \(Amazon S3\) in una directory locale](#)
- [Scarica uno stream di dimensioni sconosciute da un oggetto Amazon S3 utilizzando un AWS SDK](#)
- [Ottieni un oggetto Amazon S3 da un punto di Multi-Region accesso utilizzando un AWS SDK](#)
- [Ottieni un oggetto da un bucket Amazon S3 utilizzando un AWS SDK, specificando un'intestazione If-Modified-Since](#)
- [Inizia a utilizzare la crittografia per oggetti Amazon S3 utilizzando un AWS SDK](#)
- [Inizia a usare i tag per gli oggetti Amazon S3 utilizzando un AWS SDK](#)
- [Guida introduttiva ai cluster di elaborazione dei Big Data](#)
- [Guida introduttiva alla gestione della configurazione](#)
- [Guida introduttiva all'estrazione del testo dei documenti](#)
- [Guida introduttiva agli store di funzionalità di machine learning](#)
- [Guida introduttiva allo storage di oggetti](#)
- [Guida introduttiva all'analisi delle query](#)
- [Lavora con le funzionalità di blocco degli oggetti di Amazon S3 utilizzando un AWS SDK](#)

- [Effettua richieste condizionali per Amazon S3 utilizzando un AWS SDK](#)
- [Gestisci gli elenchi di controllo degli accessi \(ACL\) per i bucket Amazon S3 utilizzando un AWS SDK](#)
- [Gestisci messaggi Amazon SQS di grandi dimensioni utilizzando Amazon S3 con un AWS SDK](#)
- [Gestisci oggetti Amazon S3 con versioni in batch con una funzione Lambda utilizzando un AWS SDK](#)
- [Analizza gli URI di Amazon S3 utilizzando un AWS SDK](#)
- [Esegui una copia multiparte di un oggetto Amazon S3 utilizzando un AWS SDK](#)
- [Ricevi ed elabora le notifiche degli eventi di Amazon S3 utilizzando un AWS SDK](#)
- [Salvate EXIF e altre informazioni sull'immagine utilizzando un AWS SDK](#)
- [Invia notifiche di eventi S3 ad Amazon EventBridge utilizzando un AWS SDK](#)
- [Tieni traccia del caricamento o del download di un oggetto Amazon S3 utilizzando un AWS SDK](#)
- [Trasformare i dati per l'applicazione con S3 Object Lambda](#)
- [Esempi di approcci per test unitari e di integrazione con un AWS SDK](#)
- [Caricare in modo ricorsivo una directory locale in un bucket Amazon Simple Storage Service \(Amazon S3\)](#)
- [Carica o scarica file di grandi dimensioni da e verso Amazon S3 utilizzando un AWS SDK](#)
- [Carica un flusso di dimensioni sconosciute su un oggetto Amazon S3 utilizzando un AWS SDK](#)
- [Usa i checksum per lavorare con un oggetto Amazon S3 utilizzando un AWS SDK](#)
- [Lavora con le funzionalità di integrità degli oggetti di Amazon S3 utilizzando un AWS SDK](#)
- [Lavora con oggetti con versione di Amazon S3 utilizzando un AWS SDK](#)
- [Esempi serverless per Amazon S3](#)
 - [Invocazione di una funzione Lambda da un trigger Amazon S3](#)

Esempi di base per l'utilizzo di Amazon S3 AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare le nozioni di base di Amazon Simple Storage Service con gli SDK AWS .

Esempi

- [Hello Amazon S3](#)

- [Nozioni di base](#)
 - [Scopri le nozioni di base di Amazon S3 con un AWS SDK](#)

- [Azioni per l'utilizzo di Amazon S3 AWS SDK](#)
 - [Da utilizzare AbortMultipartUploadcon un AWS SDK o CLI](#)
 - [Utilizzare CompleteMultipartUploadcon un AWS SDK o CLI](#)
 - [Utilizzare CopyObjectcon un AWS SDK o CLI](#)
 - [Utilizzare CreateBucketcon un AWS SDK o CLI](#)
 - [Utilizzare CreateMultiRegionAccessPointcon un AWS SDK](#)
 - [Utilizzare CreateMultipartUploadcon un AWS SDK o CLI](#)
 - [Utilizzare CreatePresignedPostcon un AWS SDK](#)
 - [Utilizzare DeleteBucketcon un AWS SDK o CLI](#)
 - [Utilizzare DeleteBucketAnalyticsConfiguration con una CLI](#)
 - [Utilizzare DeleteBucketCorscon un AWS SDK o CLI](#)
 - [Utilizzare DeleteBucketEncryption con una CLI](#)
 - [Utilizzare DeleteBucketInventoryConfiguration con una CLI](#)
 - [Utilizzare DeleteBucketLifecyclecon un AWS SDK o CLI](#)
 - [Utilizzare DeleteBucketMetricsConfiguration con una CLI](#)
 - [Utilizzare DeleteBucketPolicycon un AWS SDK o CLI](#)
 - [Utilizzare DeleteBucketReplication con una CLI](#)
 - [Utilizzare DeleteBucketTagging con una CLI](#)
 - [Utilizzare DeleteBucketWebsitecon un AWS SDK o CLI](#)
 - [Utilizzare DeleteObjectcon un AWS SDK o CLI](#)
 - [Utilizzare DeleteObjectTagging con una CLI](#)
 - [Utilizzare DeleteObjectscon un AWS SDK o CLI](#)
 - [Utilizzare DeletePublicAccessBlock con una CLI](#)
 - [Utilizzare GetBucketAccelerateConfiguration con una CLI](#)
 - [Utilizzare GetBucketAclcon un AWS SDK o CLI](#)
 - [Utilizzare GetBucketAnalyticsConfiguration con una CLI](#)
 - [Utilizzare GetBucketCorscon un AWS SDK o CLI](#)
 - [Utilizzare GetBucketEncryptioncon un AWS SDK o CLI](#)
 - [Utilizzare GetBucketInventoryConfiguration con una CLI](#)
 - [Utilizzare GetBucketLifecycleConfigurationcon un AWS SDK o CLI](#)

- [Utilizzare GetBucketLocation con un AWS SDK o CLI](#)
- [Utilizzare GetBucketLogging con una CLI](#)
- [Utilizzare GetBucketMetricsConfiguration con una CLI](#)
- [Utilizzare GetBucketNotification con una CLI](#)
- [Utilizzare GetBucketPolicy con un AWS SDK o CLI](#)
- [Utilizzare GetBucketPolicyStatus con una CLI](#)
- [Utilizzare GetBucketReplication con un AWS SDK o CLI](#)
- [Utilizzare GetBucketRequestPayment con una CLI](#)
- [Utilizzare GetBucketTagging con una CLI](#)
- [Utilizzare GetBucketVersioning con una CLI](#)
- [Utilizzare GetBucketWebsite con un AWS SDK o CLI](#)
- [Utilizzare GetObject con un AWS SDK o CLI](#)
- [Utilizzare GetObjectAcl con un AWS SDK o CLI](#)
- [Utilizzare GetObjectAttributes con un AWS SDK o CLI](#)
- [Utilizzare GetObjectLegalHold con un AWS SDK o CLI](#)
- [Utilizzare GetObjectLockConfiguration con un AWS SDK o CLI](#)
- [Utilizzare GetObjectRetention con un AWS SDK o CLI](#)
- [Utilizzare GetObjectTagging con una CLI](#)
- [Utilizzare GetPublicAccessBlock con una CLI](#)
- [Utilizzare HeadBucket con un AWS SDK o CLI](#)
- [Utilizzare HeadObject con un AWS SDK o CLI](#)
- [Utilizzare ListBucketAnalyticsConfigurations con una CLI](#)
- [Utilizzare ListBucketInventoryConfigurations con una CLI](#)
- [Utilizzare ListBuckets con un AWS SDK o CLI](#)
- [Utilizzare ListMultipartUploads con un AWS SDK o CLI](#)
- [Utilizzare ListObjectVersions con un AWS SDK o CLI](#)
- [Utilizzare ListObjects con una CLI](#)
- [Usa ListObjectsV2 con un AWS SDK o CLI](#)
- [Utilizzare PutBucketAccelerateConfiguration con un AWS SDK o CLI](#)
- [Utilizzare PutBucketAcl con un AWS SDK o CLI](#)

- [Utilizzare PutBucketCorscon un AWS SDK o CLI](#)
- [Utilizzare PutBucketEncryptioncon un AWS SDK o CLI](#)
- [Utilizzare PutBucketLifecycleConfigurationcon un AWS SDK o CLI](#)
- [Utilizzare PutBucketLoggingcon un AWS SDK o CLI](#)
- [Utilizzare PutBucketNotification con una CLI](#)
- [Utilizzare PutBucketNotificationConfigurationcon un AWS SDK o CLI](#)
- [Utilizzare PutBucketPolicycon un AWS SDK o CLI](#)
- [Utilizzare PutBucketReplicationcon un AWS SDK o CLI](#)
- [Utilizzare PutBucketRequestPayment con una CLI](#)
- [Utilizzare PutBucketTagging con una CLI](#)
- [Utilizzare PutBucketVersioningcon un AWS SDK o CLI](#)
- [Utilizzare PutBucketWebsitecon un AWS SDK o CLI](#)
- [Utilizzare PutObjectcon un AWS SDK o CLI](#)
- [Utilizzare PutObjectAclcon un AWS SDK o CLI](#)
- [Utilizzare PutObjectLegalHoldcon un AWS SDK o CLI](#)
- [Utilizzare PutObjectLockConfigurationcon un AWS SDK o CLI](#)
- [Utilizzare PutObjectRetentioncon un AWS SDK o CLI](#)
- [Utilizzare RestoreObjectcon un AWS SDK o CLI](#)
- [Utilizzare SelectObjectContentcon un AWS SDK o CLI](#)
- [Utilizzare UploadPartcon un AWS SDK o CLI](#)
- [Utilizzare UploadPartCopycon un AWS SDK o CLI](#)

Hello Amazon S3

Gli esempi di codice seguenti mostrano come iniziare a utilizzare Amazon S3.

.NET

SDK per .NET (v4)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>
/// Hello Amazon Simple Storage Service
/// (Amazon S3) example.
/// </summary>
public class HelloS3
{
    /// <summary>
    /// Main method to run the Hello S3 example.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        var s3Client = new AmazonS3Client();

        try
        {
            Console.WriteLine("Hello Amazon S3! Let's list your buckets:");
            Console.WriteLine(new string('-', 80));

            // Use the built-in paginator to list buckets
            var request = new ListBucketsRequest();
            var paginator = s3Client.Paginators.ListBuckets(request);

            var buckets = new List<S3Bucket>();

            await foreach (var response in paginator.Responses)
            {
                buckets.AddRange(response.Buckets);
            }

            if (buckets.Any())
```

```
        {
            Console.WriteLine($"Found {buckets.Count} S3 buckets:");
            Console.WriteLine();

            foreach (var bucket in buckets)
            {
                Console.WriteLine($"- Bucket Name: {bucket.BucketName}");
                Console.WriteLine($"  Creation Date:
{bucket.CreationDate:yyyy-MM-dd HH:mm:ss UTC}");
                Console.WriteLine();
            }
        }
        else
        {
            Console.WriteLine("No S3 buckets found in your account.");
        }

        Console.WriteLine("Hello S3 completed successfully.");
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"S3 service error occurred: {ex.Message}");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Couldn't list S3 buckets. Here's why:
{ex.Message}");
    }
}
}
```

- Per i dettagli sull'API, consulta la [ListBuckets](#) sezione AWS SDK per .NET API Reference.

C++

SDK per C++

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Codice per il file CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS s3)

# Set this project's name.
project("hello_s3")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
    libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
        "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
    for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
    may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
        ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
    hello_s3.cpp)

target_link_libraries(${PROJECT_NAME}
```

```
${AWSSDK_LINK_LIBRARIES})
```

Codice per il file origine `hello_s3.cpp`.

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <iostream>
#include <aws/core/auth/AWSCredentialsProviderChain.h>
using namespace Aws;
using namespace Aws::Auth;

/*
 * A "Hello S3" starter application which initializes an Amazon Simple Storage
 * Service (Amazon S3) client
 * and lists the Amazon S3 buckets in the selected region.
 *
 * main function
 *
 * Usage: 'hello_s3'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        // You don't normally have to test that you are authenticated. But the
        // S3 service permits anonymous requests, thus the s3Client will return "success"
        // and 0 buckets even if you are unauthenticated, which can be confusing to a new
        // user.

        auto provider =
            Aws::MakeShared<DefaultAWSCredentialsProviderChain>("alloc-tag");
        auto creds = provider->GetAWSCredentials();
        if (creds.IsEmpty()) {
            std::cerr << "Failed authentication" << std::endl;
        }
    }
}
```

```

    }

    Aws::S3::S3Client s3Client(clientConfig);
    auto outcome = s3Client.ListBuckets();

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() <<
std::endl;
        result = 1;
    } else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size()
            << " buckets\n";
        for (auto &bucket: outcome.GetResult().GetBuckets()) {
            std::cout << bucket.GetName() << std::endl;
        }
    }
}

Aws::ShutdownAPI(options); // Should only be called once.
return result;
}

```

- Per i dettagli sull'API, vedi [ListBuckets](#) in AWS SDK per C++ API Reference.

Go

SDK per Go V2

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

package main

import (
    "context"
    "errors"
    "fmt"

```

```
"github.com/aws/aws-sdk-go-v2/config"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/smithy-go"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    s3Client := s3.NewFromConfig(sdkConfig)
    count := 10
    fmt.Printf("Let's list up to %v buckets for your account.\n", count)
    result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
    if err != nil {
        var ae smithy.APIError
        if errors.As(err, &ae) && ae.ErrorCode() == "AccessDenied" {
            fmt.Println("You don't have permission to list buckets for this account.")
        } else {
            fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
        }
        return
    }
    if len(result.Buckets) == 0 {
        fmt.Println("You don't have any buckets!")
    } else {
        if count > len(result.Buckets) {
            count = len(result.Buckets)
        }
        for _, bucket := range result.Buckets[:count] {
            fmt.Printf("\t%v\n", *bucket.Name)
        }
    }
}
```

- Per i dettagli sull'API, consulta la [ListBuckets](#) sezione AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBuckets(s3);
    }

    /**
```



```

    * Lists all the S3 buckets associated with the provided AWS S3 client.
    *
    * @param s3 the S3Client instance used to interact with the AWS S3 service
    */
    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Per i dettagli sull'API, consulta la [ListBuckets](#) sezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import {
    paginateListBuckets,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * List the S3 buckets in your configured AWS account.
 */
export const helloS3 = async () => {

```

```
// When no region or credentials are provided, the SDK will use the
// region and credentials from the local AWS config.
const client = new S3Client({});

try {
  /**
   * @type { import("@aws-sdk/client-s3").Bucket[] }
   */
  const buckets = [];

  for await (const page of paginateListBuckets({ client }, {})) {
    buckets.push(...page.Buckets);
  }
  console.log("Buckets: ");
  console.log(buckets.map((bucket) => bucket.Name).join("\n"));
  return buckets;
} catch (caught) {
  // ListBuckets does not throw any modeled errors. Any error caught
  // here will be something generic like `AccessDenied`.
  if (caught instanceof S3ServiceException) {
    console.error(`${caught.name}: ${caught.message}`);
  } else {
    // Something besides S3 failed.
    throw caught;
  }
}
};
```

- Per i dettagli sull'API, consulta la [ListBuckets](#) sezione AWS SDK per JavaScript API Reference.

PHP

SDK per PHP

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
use Aws\S3\S3Client;

$client = new S3Client(['region' => 'us-west-2']);
$results = $client->listBuckets();
var_dump($results);
```

- Per i dettagli sull'API, consulta la [ListBuckets](#) sezione AWS SDK per PHP API Reference.

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import boto3

def hello_s3():
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Simple Storage Service
    (Amazon S3) client and list the buckets in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """

    # Create an S3 client.
    s3_client = boto3.client("s3")

    print("Hello, Amazon S3! Let's list your buckets:")

    # Create a paginator for the list_buckets operation.
    paginator = s3_client.get_paginator("list_buckets")

    # Use the paginator to get a list of all buckets.
    response_iterator = paginator.paginate(
        PaginationConfig={
```

```

        "PageSize": 50, # Adjust PageSize as needed.
        "StartingToken": None,
    }
)

# Iterate through the pages of the response.
buckets_found = False
for page in response_iterator:
    if "Buckets" in page and page["Buckets"]:
        buckets_found = True
        for bucket in page["Buckets"]:
            print(f"\t{bucket['Name']}")

if not buckets_found:
    print("No buckets found!")

if __name__ == "__main__":
    hello_s3()

```

- Per i dettagli sull'API, consulta [ListBuckets AWS SDK for Python \(Boto3\) API Reference](#).

Ruby

SDK per Ruby

Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

# frozen_string_literal: true

# S3Manager is a class responsible for managing S3 operations
# such as listing all S3 buckets in the current AWS account.
class S3Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end
end

```

```
end

# Lists and prints all S3 buckets in the current AWS account.
def list_buckets
  @logger.info('Here are the buckets in your account:')

  response = @client.list_buckets

  if response.buckets.empty?
    @logger.info("You don't have any S3 buckets yet.")
  else
    response.buckets.each do |bucket|
      @logger.info("- #{bucket.name}")
    end
  end
rescue Aws::Errors::ServiceError => e
  @logger.error("Encountered an error while listing buckets: #{e.message}")
end
end

if $PROGRAM_NAME == __FILE__
  s3_client = Aws::S3::Client.new
  manager = S3Manager.new(s3_client)
  manager.list_buckets
end
```

- Per i dettagli sull'API, consulta la [ListBuckets](#) sezione AWS SDK per Ruby API Reference.

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// S3 Hello World Example using the AWS SDK for Rust.
///
```

```

/// This example lists the objects in a bucket, uploads an object to that bucket,
/// and then retrieves the object and prints some S3 information about the
/// object.
/// This shows a number of S3 features, including how to use built-in paginators
/// for large data sets.
///
/// # Arguments
///
/// * `client` - an S3 client configured appropriately for the environment.
/// * `bucket` - the bucket name that the object will be uploaded to. Must be
///   present in the region the `client` is configured to use.
/// * `filename` - a reference to a path that will be read and uploaded to S3.
/// * `key` - the string key that the object will be uploaded as inside the
///   bucket.
async fn list_bucket_and_upload_object(
    client: &aws_sdk_s3::Client,
    bucket: &str,
    filepath: &Path,
    key: &str,
) -> Result<(), S3ExampleError> {
    // List the buckets in this account
    let mut objects = client
        .list_objects_v2()
        .bucket(bucket)
        .into_paginator()
        .send();

    println!("key\tetag\tlast_modified\tstorage_class");
    while let Some(Ok(object)) = objects.next().await {
        for item in object.contents() {
            println!(
                "{}\t{}\t{}\t{}",
                item.key().unwrap_or_default(),
                item.e_tag().unwrap_or_default(),
                item.last_modified()
                    .map(|lm| format!("{lm}"))
                    .unwrap_or_default(),
                item.storage_class()
                    .map(|sc| format!("{sc}"))
                    .unwrap_or_default()
            );
        }
    }
}

```

```

    // Prepare a ByteStream around the file, and upload the object using that
    ByteStream.
    let body = aws_sdk_s3::primitives::ByteStream::from_path(filepath)
        .await
        .map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to create bytestream for {filepath:?} ({err:?})"
            ))
        })?;
    let resp = client
        .put_object()
        .bucket(bucket)
        .key(key)
        .body(body)
        .send()
        .await?;

    println!(
        "Upload success. Version: {:?}" ,
        resp.version_id()
        .expect("S3 Object upload missing version ID")
    );

    // Retrieve the just-uploaded object.
    let resp = client.get_object().bucket(bucket).key(key).send().await?;
    println!("etag: {}", resp.e_tag().unwrap_or("(missing)"));
    println!("version: {}", resp.version_id().unwrap_or("(missing)"));

    Ok(())
}

```

S3ExampleError utenze.

```

/// S3ExampleError provides a From<T: ProvideErrorMetadata> impl to extract
/// client-specific error details. This serves as a consistent backup to handling
/// specific service errors, depending on what is needed by the scenario.
/// It is used throughout the code examples for the AWS SDK for Rust.
#[derive(Debug)]
pub struct S3ExampleError(String);
impl S3ExampleError {
    pub fn new(value: impl Into<String>) -> Self {
        S3ExampleError(value.into())
    }
}

```

```

    }

    pub fn add_message(self, message: impl Into<String>) -> Self {
        S3ExampleError(format!("{}", message.into()), self.0))
    }
}

impl<T: aws_sdk_s3::error::ProvideErrorMetadata> From<T> for S3ExampleError {
    fn from(value: T) -> Self {
        S3ExampleError(format!(
            "{}: {}",
            value
                .code()
                .map(String::from)
                .unwrap_or("unknown code".into()),
            value
                .message()
                .map(String::from)
                .unwrap_or("missing reason".into()),
        ))
    }
}

impl std::error::Error for S3ExampleError {}

impl std::fmt::Display for S3ExampleError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
}

```

- Per i dettagli sulle API, consulta il riferimento [ListBuckets](#) all'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Scopri le nozioni di base di Amazon S3 con un AWS SDK

Gli esempi di codice seguenti mostrano come:

- Crea un bucket e carica un file in tale bucket.
- Scaricare un oggetto da un bucket.
- Copiare un oggetto in una sottocartella in un bucket.
- Elencare gli oggetti in un bucket.
- Elimina il bucket e tutti gli oggetti in esso contenuti.

.NET

SDK per .NET (v4)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegue uno scenario interattivo che dimostra le funzionalità di Amazon S3.

```
public class S3_Basics
{
    public static bool IsInteractive = true;
    public static string BucketName = null!;
    public static string TempFilePath = null!;
    public static S3Wrapper _s3Wrapper = null!;
    public static ILogger<S3_Basics> _logger = null!;

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
            {
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3Wrapper>()
                    .AddLogging(builder => builder.AddConsole());
            })
            .Build();

        _logger = LoggerFactory.Create(builder => builder.AddConsole())
            .CreateLogger<S3_Basics>();

        _s3Wrapper = host.Services.GetRequiredService<S3Wrapper>();
    }
}
```

```
var sepBar = new string('-', 45);

Console.WriteLine(sepBar);
Console.WriteLine("Amazon Simple Storage Service (Amazon S3) basic");
Console.WriteLine("procedures. This application will:");
Console.WriteLine("\n\t1. Create a bucket");
Console.WriteLine("\n\t2. Upload an object to the new bucket");
Console.WriteLine("\n\t3. Copy the uploaded object to a folder in the
bucket");
Console.WriteLine("\n\t4. List the items in the new bucket");
Console.WriteLine("\n\t5. Delete all the items in the bucket");
Console.WriteLine("\n\t6. Delete the bucket");
Console.WriteLine(sepBar);

await RunScenario(_s3Wrapper, _logger);

Console.WriteLine(sepBar);
Console.WriteLine("The Amazon S3 scenario has successfully completed.");
Console.WriteLine(sepBar);
}

/// <summary>
/// Run the S3 Basics scenario with injected dependencies.
/// </summary>
/// <param name="s3Wrapper">The S3 wrapper instance.</param>
/// <param name="scenarioLogger">The logger instance.</param>
/// <returns>A Task object.</returns>
public static async Task RunScenario(S3Wrapper s3Wrapper, ILogger<S3_Basics>
scenarioLogger)
{
    string bucketName = BucketName;
    string filePath = TempFilePath;
    string keyName = string.Empty;

    var sepBar = new string('-', 45);

    try
    {
        // Create a bucket.
        Console.WriteLine($"{sepBar}");
        Console.WriteLine("\nCreate a new Amazon S3 bucket.\n");
        Console.WriteLine(sepBar);
    }
}
```

```
        if (IsInteractive)
        {
            Console.Write("Please enter a name for the new bucket: ");
            bucketName = Console.ReadLine();
        }
        else
        {
            Console.WriteLine($"Using bucket name: {bucketName}");
        }

        var success = await s3Wrapper.CreateBucketAsync(bucketName);
        if (success)
        {
            Console.WriteLine($"Successfully created bucket: {bucketName}.
\n");
        }
        else
        {
            Console.WriteLine($"Could not create bucket: {bucketName}.\n");
        }

        Console.WriteLine(sepBar);
        Console.WriteLine("Upload a file to the new bucket.");
        Console.WriteLine(sepBar);

        if (IsInteractive)
        {
            // Get the local path and filename for the file to upload.
            while (string.IsNullOrEmpty(filePath))
            {
                Console.Write("Please enter the path and filename of the file
to upload: ");
                filePath = Console.ReadLine();

                // Confirm that the file exists on the local computer.
                if (!File.Exists(filePath))
                {
                    Console.WriteLine($"Couldn't find {filePath}. Try again.
\n");
                    filePath = string.Empty;
                }
            }
        }
        else
```

```
{
    // Use the public variable if set, otherwise create a temp file
    if (!string.IsNullOrEmpty(TempFilePath))
    {
        filePath = TempFilePath;
        Console.WriteLine($"Using provided test file: {filePath}");
    }
    else
    {
        // Create a temporary test file for non-interactive mode
        filePath = Path.GetTempFileName();
        var testContent = "This is a test file for S3 basics
scenario.\nGenerated on: " + DateTime.UtcNow.ToString("yyyy-MM-dd HH:mm:ss
UTC");

        await File.WriteAllTextAsync(filePath, testContent);
        Console.WriteLine($"Created temporary test file:
{filePath}");
    }
}

// Get the file name from the full path.
keyName = Path.GetFileName(filePath);

success = await s3Wrapper.UploadFileAsync(bucketName, keyName,
filePath);

if (success)
{
    Console.WriteLine($"Successfully uploaded {keyName} from
{filePath} to {bucketName}.\n");
}
else
{
    Console.WriteLine($"Could not upload {keyName}.\n");
}

// Set up download path
string downloadPath = string.Empty;

if (IsInteractive)
{
    // Now get a new location where we can save the file.
    while (string.IsNullOrEmpty(downloadPath))
    {
```

```
        // First get the path to which the file will be downloaded.
        Console.WriteLine("Please enter the path where the file will be
downloaded: ");

        downloadPath = Console.ReadLine();

        // Confirm that the file doesn't already exist on the local
computer.
        if (File.Exists($"{downloadPath}\\{keyName}"))
        {
            Console.WriteLine($"Sorry, the file already exists in
that location.\n");
            downloadPath = string.Empty;
        }
    }
    else
    {
        downloadPath = Path.GetTempPath();
        var downloadFile = Path.Combine(downloadPath, keyName);
        if (File.Exists(downloadFile))
        {
            File.Delete(downloadFile);
        }

        Console.WriteLine($"Using download path: {downloadPath}");
    }

    // Download an object from a bucket.
    success = await s3Wrapper.DownloadObjectFromBucketAsync(bucketName,
keyName, downloadPath);

    if (success)
    {
        Console.WriteLine($"Successfully downloaded {keyName}.\n");
    }
    else
    {
        Console.WriteLine($"Sorry, could not download {keyName}.\n");
    }

    // Copy the object to a different folder in the bucket.
    string folderName = string.Empty;

    if (IsInteractive)
```

```
{
    while (string.IsNullOrEmpty(folderName))
    {
        Console.WriteLine("Please enter the name of the folder to copy
your object to: ");
        folderName = Console.ReadLine();
    }
}
else
{
    folderName = "test-folder";
    Console.WriteLine($"Using folder name: {folderName}");
}

await s3Wrapper.CopyObjectInBucketAsync(bucketName, keyName,
folderName);

// List the objects in the bucket.
await s3Wrapper.ListBucketContentsAsync(bucketName);

// Delete the contents of the bucket.
if (IsInteractive)
{
    Console.WriteLine("Press <Enter> when you are ready to delete the
bucket contents.");
    _ = Console.ReadLine();
}

var deleteContentsSuccess = await
s3Wrapper.DeleteBucketContentsAsync(bucketName);
if (deleteContentsSuccess)
{
    Console.WriteLine($"Successfully deleted contents of
{bucketName}.\n");
}
else
{
    Console.WriteLine($"Sorry, could not delete contents of
{bucketName}.\n");
}

if (IsInteractive)
{
```

```
        // Deleting the bucket too quickly after separately deleting its
contents can
        // cause an error that the bucket isn't empty. To delete contents
and bucket in one
        // operation, use AmazonS3Util.DeleteS3BucketWithObjectsAsync
        Console.WriteLine("Press <Enter> when you are ready to delete the
bucket.");
        _ = Console.ReadLine();
    }
    else
    {
        // Add a small delay for non-interactive mode to ensure objects
are fully deleted.
        Console.WriteLine("Waiting a moment for objects to be fully
deleted...");
        await Task.Delay(2000);
    }

    // Delete the bucket.
    var deleteSuccess = await s3Wrapper.DeleteBucketAsync(bucketName);
    if (deleteSuccess)
    {
        Console.WriteLine($"Successfully deleted {bucketName}.\n");
    }
    else
    {
        Console.WriteLine($"Sorry, could not delete {bucketName}.\n");
    }

    // Clean up temporary files in non-interactive mode
    if (!IsInteractive)
    {
        try
        {
            if (File.Exists(filePath))
            {
                File.Delete(filePath);
                Console.WriteLine("Cleaned up temporary test file.");
            }

            var downloadFile = Path.Combine(downloadPath, keyName);
            if (File.Exists(downloadFile))
            {
                File.Delete(downloadFile);
            }
        }
        catch { }
    }
}
```

```
        Console.WriteLine("Cleaned up downloaded test file.");
    }
}
catch (Exception ex)
{
    scenarioLogger.LogWarning(ex, "Failed to clean up temporary
files.");
}
}
}
catch (Exception ex)
{
    scenarioLogger.LogError(ex, "An error occurred during the S3 scenario
execution.");

    // Clean up on error - delete bucket if it exists
    try
    {
        if (!string.IsNullOrEmpty(bucketName))
        {
            await s3Wrapper.DeleteBucketContentsAsync(bucketName);
            await s3Wrapper.DeleteBucketAsync(bucketName);
        }
    }
    catch (Exception cleanupEx)
    {
        scenarioLogger.LogError(cleanupEx, "Error during cleanup.");
    }

    // Clean up temporary files in non-interactive mode
    if (!IsInteractive)
    {
        try
        {
            if (!string.IsNullOrEmpty(filePath) && File.Exists(filePath))
            {
                File.Delete(filePath);
            }
        }
        catch (Exception fileCleanupEx)
        {
            scenarioLogger.LogWarning(fileCleanupEx, "Failed to clean up
temporary files during error handling.");
        }
    }
}
```



```

        }

        throw;
    }
}
}

```

Una classe wrapper per i metodi dell'SDK di Amazon S3.

```

using Amazon.S3;
using Amazon.S3.Model;

namespace S3_Actions;

/// <summary>
/// This class contains all of the methods for working with Amazon Simple
/// Storage Service (Amazon S3) buckets.
/// </summary>
public class S3Wrapper
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Initializes a new instance of the <see cref="S3Wrapper"/> class.
    /// </summary>
    /// <param name="amazonS3">An initialized Amazon S3 client object.</param>
    public S3Wrapper(IAmazonS3 amazonS3)
    {
        _amazonS3 = amazonS3;
    }

    /// <summary>
    /// Shows how to create a new Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <returns>A boolean value representing the success or failure of
    /// the bucket creation process.</returns>
    public async Task<bool> CreateBucketAsync(string bucketName)
    {
        try

```

```
{
    var request = new PutBucketRequest
    {
        BucketName = bucketName,
        UseClientRegion = true,
    };

    var response = await _amazonS3.PutBucketAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error creating bucket: '{ex.Message}'");
    return false;
}
}
```

```
/// <summary>
/// Shows how to upload a file from the local computer to an Amazon S3
/// bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to which the object
/// will be uploaded.</param>
/// <param name="objectName">The object to upload.</param>
/// <param name="filePath">The path, including file name, of the object
/// on the local computer to upload.</param>
/// <returns>A boolean value indicating the success or failure of the
/// upload procedure.</returns>
public async Task<bool> UploadFileAsync(
    string bucketName,
    string objectName,
    string filePath)
{
    try
    {
        var request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
            FilePath = filePath,
        };
    }
```

```
        var response = await _amazonS3.PutObjectAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error uploading {objectName}: {ex.Message}");
        return false;
    }
}

/// <summary>
/// Shows how to download an object from an Amazon S3 bucket to the
/// local computer.
/// </summary>
/// <param name="bucketName">The name of the bucket where the object is
/// currently stored.</param>
/// <param name="objectName">The name of the object to download.</param>
/// <param name="filePath">The path, including filename, where the
/// downloaded object will be stored.</param>
/// <returns>A boolean value indicating the success or failure of the
/// download process.</returns>
public async Task<bool> DownloadObjectFromBucketAsync(
    string bucketName,
    string objectName,
    string filePath)
{
    var request = new GetObjectRequest
    {
        BucketName = bucketName,
        Key = objectName,
    };

    using GetObjectResponse response = await
        _amazonS3.GetObjectAsync(request);

    try
    {
        // Save object to local file
        await response.WriteResponseStreamToFileAsync($"{filePath}\
        \{objectName}", true, CancellationToken.None);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
}
```

```
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error saving {objectName}: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Copies an object in an Amazon S3 bucket to a folder within the
    /// same bucket.
    /// </summary>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
    /// object to copy is located.</param>
    /// <param name="objectName">The object to be copied.</param>
    /// <param name="folderName">The folder to which the object will
    /// be copied.</param>
    /// <returns>A boolean value that indicates the success or failure of
    /// the copy operation.</returns>
    public async Task<bool> CopyObjectInBucketAsync(
        string bucketName,
        string objectName,
        string folderName)
    {
        try
        {
            var request = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = objectName,
                DestinationBucket = bucketName,
                DestinationKey = $"{folderName}\\{objectName}",
            };
            var response = await _amazonS3.CopyObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error copying object: '{ex.Message}'");
            return false;
        }
    }
}
```

```
/// <summary>
/// Shows how to list the objects in an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket for which to list.
/// <param name="printList">True to print out the list.
/// <returns>The collection of objects.</returns>
public async Task<List<S3Object>?> ListBucketContentsAsync(string bucketName,
bool printList = true)
{
    try
    {
        var request = new ListObjectsV2Request
        {
            BucketName = bucketName,
            MaxKeys = 5,
        };

        if (printList)
        {
            Console.WriteLine("-----");
            Console.WriteLine($"Listing the contents of {bucketName}:");
            Console.WriteLine("-----");
        }

        var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
        {
            BucketName = bucketName,
        });
        var s3objects = new List<S3Object>();
        await foreach (var response in listObjectsV2Paginator.Responses)
        {
            if (response.S3objects != null)
            {
                s3objects.AddRange(response.S3objects);
            }
        }

        if (printList)
        {
            Console.WriteLine($"Number of Objects: {s3objects.Count}");
            foreach (var entry in s3objects)
```

```

        {
            Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
        }
    }

    return s3objects;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error encountered on server.
Message: '{ex.Message}' getting list of objects.");
    return null;
}
}

/// <summary>
/// Delete all of the objects stored in an existing Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket from which the
/// contents will be deleted.</param>
/// <returns>A boolean value that represents the success or failure of
/// deleting all of the objects in the bucket.</returns>
public async Task<bool> DeleteBucketContentsAsync(string bucketName)
{
    // Iterate over the contents of the bucket and delete all objects.
    try
    {
        // Delete all objects in the bucket.
        var deleteList = await ListBucketContentsAsync(bucketName, false);
        if (deleteList != null && deleteList.Any())
        {
            await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
            {
                BucketName = bucketName,
                Objects = deleteList.Select(o => new KeyVersion { Key =
o.Key }).ToList(),
            });
        }

        return true;
    }
    catch (AmazonS3Exception ex)

```

```
        {
            Console.WriteLine($"Error deleting objects: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Shows how to delete an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the Amazon S3 bucket to delete.</
param>
    /// <returns>A boolean value that represents the success or failure of
    /// the delete operation.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
        try
        {
            var request = new DeleteBucketRequest { BucketName = bucketName, };

            await _amazonS3.DeleteBucketAsync(request);
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error deleting bucket: {ex.Message}");
            return false;
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per .NET .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)

- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Bash

AWS CLI con lo script Bash

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
#####
# function s3_getting_started
#
# This function creates, copies, and deletes S3 buckets and objects.
#
# Returns:
#     0 - If successful.
#     1 - If an error occurred.
#####
function s3_getting_started() {
{
    if [ "$BUCKET_OPERATIONS_SOURCED" != "True" ]; then
        cd bucket-lifecycle-operations || exit

        source ./bucket_operations.sh
        cd ..
    fi
}

echo_repeat "*" 88
echo "Welcome to the Amazon S3 getting started demo."
echo_repeat "*" 88
    echo "A unique bucket will be created by appending a Universally Unique
Identifier to a bucket name prefix."
    echo -n "Enter a prefix for the S3 bucket that will be used in this demo: "
    get_input
    bucket_name_prefix=$get_input_result
```



```
local bucket_name
bucket_name=$(generate_random_name "$bucket_name_prefix")

local region_code
region_code=$(aws configure get region)

if create_bucket -b "$bucket_name" -r "$region_code"; then
    echo "Created demo bucket named $bucket_name"
else
    errecho "The bucket failed to create. This demo will exit."
    return 1
fi

local file_name
while [ -z "$file_name" ]; do
    echo -n "Enter a file you want to upload to your bucket: "
    get_input
    file_name=$get_input_result

    if [ ! -f "$file_name" ]; then
        echo "Could not find file $file_name. Are you sure it exists?"
        file_name=""
    fi
done

local key
key="$(basename "$file_name")"

local result=0
if copy_file_to_bucket "$bucket_name" "$file_name" "$key"; then
    echo "Uploaded file $file_name into bucket $bucket_name with key $key."
else
    result=1
fi

local destination_file
destination_file="$file_name.download"
if yes_no_input "Would you like to download $key to the file $destination_file?
(y/n) "; then
    if download_object_from_bucket "$bucket_name" "$destination_file" "$key";
then
        echo "Downloaded $key in the bucket $bucket_name to the file
$destination_file."
    else
```

```
        result=1
    fi
fi

if yes_no_input "Would you like to copy $key a new object key in your bucket?
(y/n) "; then
    local to_key
    to_key="demo/$key"
    if copy_item_in_bucket "$bucket_name" "$key" "$to_key"; then
        echo "Copied $key in the bucket $bucket_name to the $to_key."
    else
        result=1
    fi
fi

local bucket_items
bucket_items=$(list_items_in_bucket "$bucket_name")

# shellcheck disable=SC2181
if [[ $? -ne 0 ]]; then
    result=1
fi

echo "Your bucket contains the following items."
echo -e "Name\t\tSize"
echo "$bucket_items"

if yes_no_input "Delete the bucket, $bucket_name, as well as the objects in it?
(y/n) "; then
    bucket_items=$(echo "$bucket_items" | cut -f 1)

    if delete_items_in_bucket "$bucket_name" "$bucket_items"; then
        echo "The following items were deleted from the bucket $bucket_name"
        echo "$bucket_items"
    else
        result=1
    fi

    if delete_bucket "$bucket_name"; then
        echo "Deleted the bucket $bucket_name"
    else
        result=1
    fi
fi
```

```

    return $result
}

```

Funzioni Amazon S3 utilizzate in questo scenario.

```

#####
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#     -b bucket_name  -- The name of the bucket to create.
#     -r region_code  -- The code for an AWS Region in which to
#                        create the bucket.
#
# Returns:
#     The URL of the bucket that was created.
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function create_bucket() {
    local bucket_name region_code response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function create_bucket"
        echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
        echo "  -b bucket_name    The name of the bucket. It must be globally
unique."
        echo "  [-r region_code]  The code for an AWS Region in which the bucket is
created."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "b:r:h" option; do
        case "${option}" in
            b) bucket_name="${OPTARG}" ;;

```

```

        r) region_code="${OPTARG}" ;;
    h)
        usage
        return 0
        ;;
    \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
esac
done

if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
fi

local bucket_config_arg
# A location constraint for "us-east-1" returns an error.
if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
fi

iecho "Parameters:\n"
iecho "    Bucket name:  $bucket_name"
iecho "    Region code:  $region_code"
iecho ""

# If the bucket already exists, we don't want to try to create it.
if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
fi

# shellcheck disable=SC2086
response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then

```

```

        errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
        return 1
    fi
}

#####
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file to.
#     $2 - The path and file name of the local file to copy to the bucket.
#     $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function copy_file_to_bucket() {
    local response bucket_name source_file destination_file_name
    bucket_name=$1
    source_file=$2
    destination_file_name=$3

    response=$(aws s3api put-object \
        --bucket "$bucket_name" \
        --body "$source_file" \
        --key "$destination_file_name")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "ERROR: AWS reports put-object operation failed.\n$response"
        return 1
    fi
}

#####
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#     $1 - The name of the bucket to download the object from.

```

```

#      $2 - The path and file name to store the downloaded bucket.
#      $3 - The key (name) of the object in the bucket.
#
# Returns:
#      0 - If successful.
#      1 - If it fails.
#####
function download_object_from_bucket() {
    local bucket_name=$1
    local destination_file_name=$2
    local object_name=$3
    local response

    response=$(aws s3api get-object \
        --bucket "$bucket_name" \
        --key "$object_name" \
        "$destination_file_name")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "ERROR: AWS reports put-object operation failed.\n$response"
        return 1
    fi
}

#####
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#      $1 - The name of the bucket to copy the file from and to.
#      $2 - The key of the source file to copy.
#      $3 - The key of the destination file.
#
# Returns:
#      0 - If successful.
#      1 - If it fails.
#####
function copy_item_in_bucket() {
    local bucket_name=$1
    local source_key=$2
    local destination_key=$3
    local response

```

```

response=$(aws s3api copy-object \
  --bucket "$bucket_name" \
  --copy-source "$bucket_name/$source_key" \
  --key "$destination_key")

# shellcheck disable=SC2181
if [[ $? -ne 0 ]]; then
  errecho "ERROR: AWS reports s3api copy-object operation failed.\n$response"
  return 1
fi
}

#####
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#   $1 - The name of the bucket.
#
# Returns:
#   The list of files in text format.
#   And:
#   0 - If successful.
#   1 - If it fails.
#####
function list_items_in_bucket() {
  local bucket_name=$1
  local response

  response=$(aws s3api list-objects \
    --bucket "$bucket_name" \
    --output text \
    --query 'Contents[].{Key: Key, Size: Size}')

  # shellcheck disable=SC2181
  if [[ ${?} -eq 0 ]]; then
    echo "$response"
  else
    errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
    return 1
  fi
}

```

```

    fi
}

#####
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.
#     $2 - A list of keys in the bucket to delete.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_items_in_bucket() {
    local bucket_name=$1
    local keys=$2
    local response

    # Create the JSON for the items to delete.
    local delete_items
    delete_items="{\"Objects\":["
    for key in $keys; do
        delete_items="$delete_items{\"Key\": \"$key\"},"
    done
    delete_items=${delete_items%?} # Remove the final comma.
    delete_items="$delete_items]"

    response=$(aws s3api delete-objects \
        --bucket "$bucket_name" \
        --delete "$delete_items")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-object operation failed.\n
$response"
        return 1
    fi
}

#####
# function delete_bucket

```



```
#
# This function deletes the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api delete-bucket \
        --bucket "$bucket_name")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
        return 1
    fi
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dei comandi della AWS CLI .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

C++

SDK per C++

 Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CopyObjectRequest.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/DeleteBucketRequest.h>
#include <aws/s3/model/DeleteObjectRequest.h>
#include <aws/s3/model/GetObjectRequest.h>
#include <aws/s3/model/ListObjectsV2Request.h>
#include <aws/s3/model/PutObjectRequest.h>
#include <aws/s3/model/BucketLocationConstraint.h>
#include <aws/s3/model/CreateBucketConfiguration.h>
#include <aws/core/utils/UUID.h>
#include <aws/core/utils/StringUtils.h>
#include <aws/core/utils/memory/stl/AWSAllocator.h>
#include <fstream>
#include "s3_examples.h"

namespace AwsDoc {
    namespace S3 {

        //! Delete an S3 bucket.
        /*!
         * \param bucketName: The S3 bucket's name.
         * \param client: An S3 client.
         * \return bool: Function succeeded.
         */
        static bool
        deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client);

        //! Delete an object in an S3 bucket.
        /*!
         */
    }
```

```

        \param bucketName: The S3 bucket's name.
        \param key: The key for the object in the S3 bucket.
        \param client: An S3 client.
        \return bool: Function succeeded.
    */
    static bool
    deleteObjectFromBucket(const Aws::String &bucketName, const Aws::String
&key,
                           Aws::S3::S3Client &client);
    }
}

//! Scenario to create, copy, and delete S3 buckets and objects.
/*!
    \param bucketNamePrefix: A prefix for a bucket name.
    \param uploadFilePath: Path to file to upload to an Amazon S3 bucket.
    \param saveFilePath: Path for saving a downloaded S3 object.
    \param clientConfig: Aws client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::S3_GettingStartedScenario(const Aws::String &bucketNamePrefix,
      const Aws::String &uploadFilePath,
      const Aws::String &saveFilePath,
      const Aws::Client::ClientConfiguration
&clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    // Create a unique bucket name which is only temporary and will be deleted.
    // Format: <bucketNamePrefix> + "-" + lowercase UUID.
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String bucketName = bucketNamePrefix +
        Aws::Utils::StringUtils::ToLower(uuid.c_str());

    // 1. Create a bucket.
    {
        Aws::S3::Model::CreateBucketRequest request;
        request.SetBucket(bucketName);

        if (clientConfig.region != Aws::Region::US_EAST_1) {
            Aws::S3::Model::CreateBucketConfiguration createBucketConfiguration;
            createBucketConfiguration.WithLocationConstraint(

```

```

    Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
        clientConfig.region));
    request.WithCreateBucketConfiguration(createBucketConfiguration);
}

    Aws::S3::Model::CreateBucketOutcome outcome =
client.CreateBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
        return false;
    } else {
        std::cout << "Created the bucket, '" << bucketName <<
            "', in the region, '" << clientConfig.region << "'." <<
std::endl;
    }
}

// 2. Upload a local file to the bucket.
Aws::String key = "key-for-test";
{
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    std::shared_ptr<Aws::FStream> input_data =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
            uploadFilePath,
            std::ios_base::in |
            std::ios_base::binary);

    if (!input_data->is_open()) {
        std::cerr << "Error: unable to open file, '" << uploadFilePath <<
            "'." << std::endl;
        AwsDoc::S3::deleteBucket(bucketName, client);
        return false;
    }

    request.SetBody(input_data);
}

```

```

    Aws::S3::Model::PutObjectOutcome outcome =
        client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putObject: " <<
            outcome.GetError().GetMessage() << std::endl;
        AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);
        AwsDoc::S3::deleteBucket(bucketName, client);
        return false;
    } else {
        std::cout << "Added the object with the key, '" << key
            << "', to the bucket, '"
            << bucketName << "'." << std::endl;
    }
}

// 3. Download the object to a local file.
{
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::GetObjectOutcome outcome =
        client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Downloaded the object with the key, '" << key
            << "', in the bucket, '"
            << bucketName << "'." << std::endl;

        Aws::IOStream &ioStream = outcome.GetResultWithOwnership().
            GetBody();
        Aws::OFStream outStream(saveFilePath,
                                std::ios_base::out | std::ios_base::binary);
        if (!outStream.is_open()) {
            std::cout << "Error: unable to open file, '" << saveFilePath <<
                "'."
                << std::endl;

```

```
        } else {
            outputStream << ioStream.rdbuf();
            std::cout << "Wrote the downloaded object to the file '"
                      << saveFilePath << "'." << std::endl;
        }
    }
}

// 4. Copy the object to a different "folder" in the bucket.
Aws::String copiedToKey = "test-folder/" + key;
{
    Aws::S3::Model::CopyObjectRequest request;
    request.WithBucket(bucketName)
           .WithKey(copiedToKey)
           .WithCopySource(bucketName + "/" + key);

    Aws::S3::Model::CopyObjectOutcome outcome =
        client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error: copyObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Copied the object with the key, '" << key
                  << "', to the key, '" << copiedToKey
                  << "', in the bucket, '" << bucketName << "'." << std::endl;
    }
}

// 5. List objects in the bucket.
{
    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken;
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }
        Aws::S3::Model::ListObjectsV2Outcome outcome = client.ListObjectsV2(
            request);

        if (!outcome.IsSuccess()) {
```

```

        std::cerr << "Error: ListObjects: " <<
            outcome.GetError().GetMessage() << std::endl;
        break;
    } else {
        Aws::Vector<Aws::S3::Model::Object> objects =
            outcome.GetResult().GetContents();
        allObjects.insert(allObjects.end(), objects.begin(),
objects.end());
        continuationToken = outcome.GetResult().GetContinuationToken();
    }
} while (!continuationToken.empty());

std::cout << allObjects.size() << " objects in the bucket, '" <<
bucketName
    << "':" << std::endl;

for (Aws::S3::Model::Object &object: allObjects) {
    std::cout << "    '" << object.GetKey() << "'" << std::endl;
}
}

// 6. Delete all objects in the bucket.
// All objects in the bucket must be deleted before deleting the bucket.
AwsDoc::S3::deleteObjectFromBucket(bucketName, copiedToKey, client);
AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);

// 7. Delete the bucket.
return AwsDoc::S3::deleteBucket(bucketName, client);
}

bool AwsDoc::S3::deleteObjectFromBucket(const Aws::String &bucketName,
                                        const Aws::String &key,
                                        Aws::S3::S3Client &client) {
    Aws::S3::Model::DeleteObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: deleteObject: " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {

```

```

        std::cout << "Deleted the object with the key, '" << key
                    << "', from the bucket, '"
                    << bucketName << "'.'" << std::endl;
    }

    return outcome.IsSuccess();
}

bool
AwsDoc::S3::deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client
&client) {
    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
        client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                    err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Deleted the bucket, '" << bucketName << "'.'" << std::endl;
    }
    return outcome.IsSuccess();
}

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per C++ .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Go

SDK per Go V2

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Definisci una struttura che racchiude le azioni del bucket e dell'oggetto utilizzate dallo scenario.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform  
// bucket  
// and object actions.  
type BucketBasics struct {  
    S3Client *s3.Client  
}
```

```
// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket,
    error) {
    var err error
    var output *s3.ListBucketsOutput
    var buckets []types.Bucket
    bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
        &s3.ListBucketsInput{})
    for bucketPaginator.HasMorePages() {
        output, err = bucketPaginator.NextPage(ctx)
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
                fmt.Println("You don't have permission to list buckets for this account.")
                err = apiErr
            } else {
                log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
            }
            break
        } else {
            buckets = append(buckets, output.Buckets...)
        }
    }
    return buckets, err
}

// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
    (bool, error) {
    _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
        Bucket: aws.String(bucketName),
    })
    exists := true
    if err != nil {
        var apiError smithy.APIError
        if errors.As(err, &apiError) {
            switch apiError.(type) {
            case *types.NotFound:
                log.Printf("Bucket %v is available.\n", bucketName)
                exists = false
            }
            err = nil
        }
    }
}
```

```
    default:
        log.Printf("Either you don't have access to bucket %v or another error
occurred. "+
        "Here's what happened: %v\n", bucketName, err)
    }
}
} else {
    log.Printf("Bucket %v exists and you already own it.", bucketName)
}

return exists, err
}

// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
string) error {
    _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
        Bucket: aws.String(name),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    })
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", name)
            err = owned
        } else if errors.As(err, &exists) {
            log.Printf("Bucket %s already exists.\n", name)
            err = exists
        }
    } else {
        err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
        }
    }
    return err
}
```

```
// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    file, err := os.Open(fileName)
    if err != nil {
        log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
    } else {
        defer file.Close()
        _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
            Bucket: aws.String(bucketName),
            Key:     aws.String(objectKey),
            Body:    file,
        })
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
                log.Printf("Error while uploading object to %s. The object is too large.\n"+
                    "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
                    "or the multipart upload API (5TB max).", bucketName)
            } else {
                log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
                    fileName, bucketName, objectKey, err)
            }
        } else {
            err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
                    aws.String(objectKey)}, time.Minute)
            if err != nil {
                log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
            }
        }
    }
    return err
}

// UploadLargeObject uses an upload manager to upload data to an object in a
// bucket.
// The upload manager breaks large data into parts and uploads the parts
// concurrently.
```

```

func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName
    string, objectKey string, largeObject []byte) error {
    largeBuffer := bytes.NewReader(largeObject)
    var partMiBs int64 = 10
    uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
        u.PartSize = partMiBs * 1024 * 1024
    })
    _, err := uploader.Upload(ctx, &s3.PutObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
        Body:    largeBuffer,
    })
    if err != nil {
        var apiErr smithy.APIError
        if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
            log.Printf("Error while uploading object to %s. The object is too large.\n"+
                "The maximum size for a multipart upload is 5TB.", bucketName)
        } else {
            log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
                bucketName, objectKey, err)
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
                aws.String(objectKey)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
        }
    }

    return err
}

// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:    aws.String(objectKey),
    })
    if err != nil {
        var noKey *types.NoSuchKey

```

```

    if errors.As(err, &noKey) {
        log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
            objectKey, bucketName)
        err = noKey
    } else {
        log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
            objectKey, err)
    }
    return err
}
defer result.Body.Close()
file, err := os.Create(fileName)
if err != nil {
    log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
    return err
}
defer file.Close()
body, err := io.ReadAll(result.Body)
if err != nil {
    log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
        err)
}
_, err = file.Write(body)
return err
}

// DownloadLargeObject uses a download manager to download an object from a
// bucket.
// The download manager gets the data in parts and writes them to a buffer until
// all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
    string, objectKey string) ([]byte, error) {
    var partMiBs int64 = 10
    downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
    {
        d.PartSize = partMiBs * 1024 * 1024
    })
    buffer := manager.NewWriteAtBuffer([]byte{})
    _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:    aws.String(objectKey),
    })

```

```
    })
    if err != nil {
        log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return buffer.Bytes(), err
}

// CopyToFolder copies an object in a bucket to a subfolder in the same bucket.
func (basics BucketBasics) CopyToFolder(ctx context.Context, bucketName string,
    objectKey string, folderName string) error {
    objectDest := fmt.Sprintf("%v/%v", folderName, objectKey)
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
        Bucket:      aws.String(bucketName),
        CopySource:  aws.String(fmt.Sprintf("%v/%v", bucketName, objectKey)),
        Key:         aws.String(objectDest),
    })
    if err != nil {
        var notActive *types.ObjectNotInActiveTierError
        if errors.As(err, &notActive) {
            log.Printf("Couldn't copy object %s from %s because the object isn't in the
                active tier.\n",
                objectKey, bucketName)
            err = notActive
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
                aws.String(objectDest)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectDest)
        }
    }
    return err
}

// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
    destinationBucket string, objectKey string) error {
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
```

```

    Bucket:      aws.String(destinationBucket),
    CopySource:  aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
    Key:         aws.String(objectKey),
})
if err != nil {
    var notActive *types.ObjectNotInActiveTierError
    if errors.As(err, &notActive) {
        log.Printf("Couldn't copy object %s from %s because the object isn't in the
active tier.\n",
            objectKey, sourceBucket)
        err = notActive
    }
} else {
    err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
        ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
aws.String(objectKey)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
    }
}
return err
}

// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
([]types.Object, error) {
    var err error
    var output *s3.ListObjectsV2Output
    input := &s3.ListObjectsV2Input{
        Bucket: aws.String(bucketName),
    }
    var objects []types.Object
    objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
    for objectPaginator.HasMorePages() {
        output, err = objectPaginator.NextPage(ctx)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucketName)
                err = noBucket
            }
            break
        }
    }
}

```



```

    } else {
        objects = append(objects, output.Contents...)
    }
}
return objects, err
}

// DeleteObjects deletes a list of objects from a bucket.
func (basics BucketBasics) DeleteObjects(ctx context.Context, bucketName string,
    objectKeys []string) error {
    var objectIds []types.ObjectIdentifier
    for _, key := range objectKeys {
        objectIds = append(objectIds, types.ObjectIdentifier{Key: aws.String(key)})
    }
    output, err := basics.S3Client.DeleteObjects(ctx, &s3.DeleteObjectsInput{
        Bucket: aws.String(bucketName),
        Delete: &types.Delete{Objects: objectIds, Quiet: aws.Bool(true)},
    })
    if err != nil || len(output.Errors) > 0 {
        log.Printf("Error deleting objects from bucket %s.\n", bucketName)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucketName)
                err = noBucket
            }
        }
        if len(output.Errors) > 0 {
            for _, outErr := range output.Errors {
                log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
            }
            err = fmt.Errorf("%s", *output.Errors[0].Message)
        }
    } else {
        for _, delObjs := range output.Deleted {
            err = s3.NewObjectNotExistsWaiter(basics.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key: delObjs.Key},
                time.Minute)
            if err != nil {
                log.Printf("Failed attempt to wait for object %s to be deleted.\n",
                    *delObjs.Key)
            } else {
                log.Printf("Deleted %s.\n", *delObjs.Key)
            }
        }
    }
}

```

```

    }
    }
}
return err
}

// DeleteBucket deletes a bucket. The bucket must be empty or an error is
// returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
error {
    _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
        Bucket: aws.String(bucketName)})
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucketName)
            err = noBucket
        } else {
            log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
        }
    } else {
        err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
                bucketName)
        } else {
            log.Printf("Deleted %s.\n", bucketName)
        }
    }
}
return err
}

```

Esegue uno scenario interattivo che mostra come utilizzare i bucket e gli oggetti S3.

```

import (
    "context"
    "fmt"

```

```

"log"
"os"
"strings"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunGetStartedScenario is an interactive example that shows you how to use
// Amazon
// Simple Storage Service (Amazon S3) to create an S3 bucket and use it to store
// objects.
//
// 1. Create a bucket.
// 2. Upload a local file to the bucket.
// 3. Download an object to a local file.
// 4. Copy an object to a different folder in the bucket.
// 5. List objects in the bucket.
// 6. Delete all objects in the bucket.
// 7. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
// so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../../demotools folder of this repo.
func RunGetStartedScenario(ctx context.Context, sdkConfig aws.Config, questioner
demotools.IQuestioner) {
defer func() {
if r := recover(); r != nil {
log.Println("Something went wrong with the demo.")
_, isMock := questioner.(*demotools.MockQuestioner)
if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
log.Println(r)
}
}
}()

log.Println(strings.Repeat("-", 88))

```

```
log.Println("Welcome to the Amazon S3 getting started demo.")
log.Println(strings.Repeat("-", 88))

s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}

count := 10
log.Printf("Let's list up to %v buckets for your account:", count)
buckets, err := bucketBasics.ListBuckets(ctx)
if err != nil {
    panic(err)
}
if len(buckets) == 0 {
    log.Println("You don't have any buckets!")
} else {
    if count > len(buckets) {
        count = len(buckets)
    }
    for _, bucket := range buckets[:count] {
        log.Printf("\t\t%v\n", *bucket.Name)
    }
}

bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
bucket:",
    demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
    panic(err)
}
if !bucketExists {
    err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
    if err != nil {
        panic(err)
    } else {
        log.Println("Bucket created.")
    }
}

log.Println(strings.Repeat("-", 88))

fmt.Println("Let's upload a file to your bucket.")
smallFile := questioner.Ask("Enter the path to a file you want to upload:",
    demotools.NotEmpty{})
const smallKey = "doc-example-key"
```

```
err = bucketBasics.UploadFile(ctx, bucketName, smallKey, smallFile)
if err != nil {
    panic(err)
}
log.Printf("Uploaded %v as %v.\n", smallFile, smallKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download %v to a file.", smallKey)
downloadFileName := questioner.Ask("Enter a name for the downloaded file:",
demotools.NotEmpty{})
err = bucketBasics.DownloadFile(ctx, bucketName, smallKey, downloadFileName)
if err != nil {
    panic(err)
}
log.Printf("File %v downloaded.", downloadFileName)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's copy %v to a folder in the same bucket.", smallKey)
folderName := questioner.Ask("Enter a folder name: ", demotools.NotEmpty{})
err = bucketBasics.CopyToFolder(ctx, bucketName, smallKey, folderName)
if err != nil {
    panic(err)
}
log.Printf("Copied %v to %v/%v.\n", smallKey, folderName, smallKey)
log.Println(strings.Repeat("-", 88))

log.Println("Let's list the objects in your bucket.")
questioner.Ask("Press Enter when you're ready.")
objects, err := bucketBasics.ListObjects(ctx, bucketName)
if err != nil {
    panic(err)
}
log.Printf("Found %v objects.\n", len(objects))
var objKeys []string
for _, object := range objects {
    objKeys = append(objKeys, *object.Key)
    log.Printf("\t%v\n", *object.Key)
}
log.Println(strings.Repeat("-", 88))

if questioner.AskBool("Do you want to delete your bucket and all of its "+
"contents? (y/n)", "y") {
    log.Println("Deleting objects.")
    err = bucketBasics.DeleteObjects(ctx, bucketName, objKeys)
```

```
if err != nil {
    panic(err)
}
log.Println("Deleting bucket.")
err = bucketBasics.DeleteBucket(ctx, bucketName)
if err != nil {
    panic(err)
}
log.Printf("Deleting downloaded file %v.\n", downloadFileName)
err = os.Remove(downloadFileName)
if err != nil {
    panic(err)
}
} else {
    log.Println("Okay. Don't forget to delete objects from your bucket to avoid charges.")
}
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Go .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Java

SDK per Java 2.x

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Scenario di esempio.

```
import java.io.IOException;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
 * 2. Uploads an object to the bucket.
 * 3. Downloads the object to another local file.
 * 4. Uploads an object using multipart upload.
 * 5. List all objects located in the Amazon S3 bucket.
 * 6. Copies the object to another Amazon S3 bucket.
 * 7. Copy the object to another Amazon S3 bucket using multi copy.
 * 8. Deletes the object from the Amazon S3 bucket.
 * 9. Deletes the Amazon S3 bucket.
 */
```

```

public class S3Scenario {

    public static Scanner scanner = new Scanner(System.in);
    static S3Actions s3Actions = new S3Actions();
    public static final String DASHES = new String(new char[80]).replace("\0",
"-");
    private static final Logger logger =
LoggerFactory.getLogger(S3Scenario.class);
    public static void main(String[] args) throws IOException {
        final String usage = ""
            Usage:
                <bucketName> <key> <objectPath> <savePath> <toBucket>

        Where:
            bucketName - The name of the S3 bucket.
            key - The unique identifier for the object stored in the S3
bucket.
            objectPath - The full file path of the object within the S3
bucket (e.g., "documents/reports/annual_report.pdf").
            savePath - The local file path where the object will be
downloaded and saved (e.g., "C:/Users/username/Downloads/annual_report.pdf").
            toBucket - The name of the S3 bucket to which the object will be
copied.

        """;

        if (args.length != 5) {
            logger.info(usage);
            return;
        }

        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4];

        logger.info(DASHES);
        logger.info("Welcome to the Amazon Simple Storage Service (S3) example
scenario.");
        logger.info("
            Amazon S3 is a highly scalable and durable object storage
            service provided by Amazon Web Services (AWS). It is designed to
            store and retrieve

```


any amount of data, from anywhere on the web, at any time.

The `S3AsyncClient` interface in the AWS SDK for Java 2.x provides a set of methods to programmatically interact with the Amazon S3 (Simple Storage Service) service. This allows developers to automate the management and manipulation of S3 buckets and objects as part of their application deployment pipelines. With S3, teams can focus on building and deploying their applications without having to worry about the underlying storage infrastructure required to host and manage large amounts of data.

This scenario walks you through how to perform key operations for this service.

Let's get started...

```
""");
waitForInputToContinue(scanner);
logger.info(DASHES);

try {
    // Run the methods that belong to this scenario.
    runScenario(bucketName, key, objectPath, savePath, toBucket);

} catch (Throwable rt) {
    Throwable cause = rt.getCause();
    if (cause instanceof S3Exception kmsEx) {
        logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
    } else {
        logger.info("An unexpected error occurred: " + rt.getMessage());
    }
}

private static void runScenario(String bucketName, String key, String
objectPath, String savePath, String toBucket) throws Throwable {
    logger.info(DASHES);
    logger.info("1. Create an Amazon S3 bucket.");
    try {
        CompletableFuture<Void> future =
s3Actions.createBucketAsync(bucketName);
        future.join();
    }
```

```
        waitForInputToContinue(scanner);

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code  
{})", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }

    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("2. Upload a local file to the Amazon S3 bucket.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<PutObjectResponse> future =
s3Actions.uploadLocalFileAsync(bucketName, key, objectPath);
        future.join();
        logger.info("File uploaded successfully to {}/{}", bucketName, key);

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code  
{})", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("3. Download the object to another local file.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<Void> future =
s3Actions.getObjectBytesAsync(bucketName, key, savePath);
```

```
        future.join();
        logger.info("Successfully obtained bytes from S3 object and wrote to
file {}", savePath);

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("4. Perform a multipart upload.");
    waitForInputToContinue(scanner);
    String multipartKey = "multiPartKey";
    try {
        // Call the multipartUpload method
        CompletableFuture<Void> future =
s3Actions.multipartUpload(bucketName, multipartKey);
        future.join();
        logger.info("Multipart upload completed successfully for bucket '{}'  
and key '{}'", bucketName, multipartKey);

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("5. List all objects located in the Amazon S3 bucket.");
```

```
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.listAllObjectsAsync(bucketName);
            future.join();
            logger.info("Object listing completed successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Copy the object to another Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
s3Actions.copyBucketObjectAsync(bucketName, key, toBucket);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
```

```
        logger.info("7. Copy the object to another Amazon S3 bucket using multi
copy.");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<String> future =
s3Actions.performMultiCopy(toBucket, bucketName, key);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Delete objects from the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, key);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, "multiPartKey");
```

```
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("9. Delete the Amazon S3 bucket.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<Void> future =
s3Actions.deleteBucketAsync(bucketName);
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("You successfully completed the Amazon S3 scenario.");
    logger.info(DASHES);
}

private static void waitForInputToContinue(Scanner scanner) {
    while (true) {
        logger.info("");
    }
}
```

```

        logger.info("Enter 'c' followed by <ENTER> to continue:");
        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
            logger.info("Continuing with the program...");
            logger.info("");
            break;
        } else {
            // Handle invalid input.
            logger.info("Invalid input. Please try again.");
        }
    }
}
}

```

Una classe wrapper che contiene le operazioni.

```

public class S3Actions {

    private static final Logger logger =
    LoggerFactory.getLogger(S3Actions.class);
    private static S3AsyncClient s3AsyncClient;

    public static S3AsyncClient getAsyncClient() {
        if (s3AsyncClient == null) {
            /*
            The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
            version 2,
            and it is designed to provide a high-performance, asynchronous HTTP
            client for interacting with AWS services.
            It uses the Netty framework to handle the underlying network
            communication and the Java NIO API to
            provide a non-blocking, event-driven approach to HTTP requests and
            responses.
            */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50) // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60)) // Set the connection
            timeout.

                .readTimeout(Duration.ofSeconds(60)) // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60)) // Set the write timeout.

```

```

        .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofMinutes(2)) // Set the overall API
call timeout.
        .apiCallAttemptTimeout(Duration.ofSeconds(90)) // Set the
individual call attempt timeout.
        .retryStrategy(RetryMode.STANDARD)
        .build();

        s3AsyncClient = S3AsyncClient.builder()
        .region(Region.US_EAST_1)
        .httpClient(httpClient)
        .overrideConfiguration(overrideConfig)
        .build();
    }
    return s3AsyncClient;
}

/**
 * Creates an S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to create
 * @return a {@link CompletableFuture} that completes when the bucket is
created and ready
 * @throws RuntimeException if there is a failure while creating the bucket
 */
public CompletableFuture<Void> createBucketAsync(String bucketName) {
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .build();

    CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
    return response.thenCompose(resp -> {
        S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =

```



```

        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        return waiterResponseFuture.thenAccept(waiterResponse -> {
            waiterResponse.matched().response().ifPresent(headBucketResponse
-> {
                logger.info(bucketName + " is ready");
            });
        });
    }).whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to create bucket", ex);
        }
    });
}

/**
 * Uploads a local file to an AWS S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to upload the file to
 * @param key         the key (object name) to use for the uploaded file
 * @param objectPath  the local file path of the file to be uploaded
 * @return a {@link CompletableFuture} that completes with the {@link
PutObjectResponse} when the upload is successful, or throws a {@link
RuntimeException} if the upload fails
 */
public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
    return response.whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to upload file", ex);
        }
    });
}

/**

```

```

    * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket
    and writes them to a local file.
    *
    * @param bucketName the name of the S3 bucket containing the object
    * @param keyName    the key (or name) of the S3 object to retrieve
    * @param path       the local file path where the object's bytes will be
    written
    * @return a {@link CompletableFuture} that completes when the object bytes
    have been written to the local file
    */
    public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
    keyName, String path) {
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        CompletableFuture<ResponseBytes<GetObjectResponse>> response =
        getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
        return response.thenAccept(objectBytes -> {
            try {
                byte[] data = objectBytes.asByteArray();
                Path filePath = Paths.get(path);
                Files.write(filePath, data);
                logger.info("Successfully obtained bytes from an S3 object");
            } catch (IOException ex) {
                throw new RuntimeException("Failed to write data to file", ex);
            }
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get object bytes from S3",
ex);
            }
        });
    }

    /**
    * Asynchronously lists all objects in the specified S3 bucket.
    *
    * @param bucketName the name of the S3 bucket to list objects for
    * @return a {@link CompletableFuture} that completes when all objects have
    been listed
    */

```

```

    public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
        ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Publisher paginator =
getAsyncClient().listObjectsV2Paginator(initialRequest);
        return paginator.subscribe(response -> {
            response.contents().forEach(s3Object -> {
                logger.info("Object key: " + s3Object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
bucketName);
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }

    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param fromBucket the name of the source S3 bucket
     * @param objectKey the key (name) of the object to be copied
     * @param toBucket the name of the destination S3 bucket
     * @return a {@link CompletableFuture} that completes with the copy result as
a {@link String}
     * @throws RuntimeException if the URL could not be encoded or an S3
exception occurred during the copy
     */
    public CompletableFuture<String> copyBucketObjectAsync(String fromBucket,
String objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(fromBucket)
            .sourceKey(objectKey)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();

        CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
        response.whenComplete((copyRes, ex) -> {

```

```

        if (copyRes != null) {
            logger.info("The " + objectKey + " was copied to " + toBucket);
        } else {
            throw new RuntimeException("An S3 exception occurred during
copy", ex);
        }
    });

    return response.thenApply(CopyObjectResponse::copyObjectResult)
        .thenApply(Object::toString);
}

/**
 * Performs a multipart upload to an Amazon S3 bucket.
 *
 * @param bucketName the name of the S3 bucket to upload the file to
 * @param key         the key (name) of the file to be uploaded
 * @return a {@link CompletableFuture} that completes when the multipart
upload is successful
 */
public CompletableFuture<Void> multipartUpload(String bucketName, String key)
{
    int mB = 1024 * 1024;

    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    return
getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
        .thenCompose(createResponse -> {
            String uploadId = createResponse.uploadId();
            System.out.println("Upload ID: " + uploadId);

            // Upload part 1.
            UploadPartRequest uploadPartRequest1 =
UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .partNumber(1)
                .contentLength((long) (5 * mB)) // Specify the content length

```

```
        .build());

        CompletableFuture<CompletedPart> part1Future =
getAsyncClient().uploadPart(uploadPartRequest1,
        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(5 *
mB)))

        .thenApply(uploadPartResponse -> CompletedPart.builder()
        .partNumber(1)
        .eTag(uploadPartResponse.eTag())
        .build());

        // Upload part 2.
        UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder()
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)
        .partNumber(2)
        .contentLength((long) (3 * mB))
        .build();

        CompletableFuture<CompletedPart> part2Future =
getAsyncClient().uploadPart(uploadPartRequest2,
        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(3 *
mB)))

        .thenApply(uploadPartResponse -> CompletedPart.builder()
        .partNumber(2)
        .eTag(uploadPartResponse.eTag())
        .build());

        // Combine the results of both parts.
        return CompletableFuture.allOf(part1Future, part2Future)
        .thenCompose(v -> {
            CompletedPart part1 = part1Future.join();
            CompletedPart part2 = part2Future.join();

            CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                .parts(part1, part2)
                .build();

            CompleteMultipartUploadRequest
completeMultipartUploadRequest = CompleteMultipartUploadRequest.builder()
                .bucket(bucketName)
```

```

        .key(key)
        .uploadId(uploadId)
        .multipartUpload(completedMultipartUpload)
        .build();

        // Complete the multipart upload
        return
getAsyncClient().completeMultipartUpload(completeMultipartUploadRequest);
    });
})
    .thenAccept(response -> System.out.println("Multipart upload
completed successfully"))
    .exceptionally(ex -> {
        System.err.println("Failed to complete multipart upload: " +
ex.getMessage());
        throw new RuntimeException(ex);
    });
}

/**
 * Deletes an object from an S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the key (file name) of the object to be deleted
 * @return a {@link CompletableFuture} that completes when the object has
been deleted
 */
public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
    response.whenComplete((deleteRes, ex) -> {
        if (deleteRes != null) {
            logger.info(key + " was deleted");
        } else {
            throw new RuntimeException("An S3 exception occurred during
delete", ex);
        }
    })
}

```

```
    });

    return response.thenApply(r -> null);
}

/**
 * Deletes an S3 bucket asynchronously.
 *
 * @param bucket the name of the bucket to be deleted
 * @return a {@link CompletableFuture} that completes when the bucket
deletion is successful, or throws a {@link RuntimeException}
 * if an error occurs during the deletion process
 */
public CompletableFuture<Void> deleteBucketAsync(String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();

    CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
    response.whenComplete((deleteRes, ex) -> {
        if (deleteRes != null) {
            logger.info(bucket + " was deleted.");
        } else {
            throw new RuntimeException("An S3 exception occurred during
bucket deletion", ex);
        }
    });
    return response.thenApply(r -> null);
}

public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
        .bucket(toBucket)
        .key(key)
        .build();

    getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
        .thenApply(createMultipartUploadResponse -> {
            String uploadId = createMultipartUploadResponse.uploadId();
            System.out.println("Upload ID: " + uploadId);
        });
}
```

```

        UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
    .sourceBucket(bucketName)
    .destinationBucket(toBucket)
    .sourceKey(key)
    .destinationKey(key)
    .uploadId(uploadId) // Use the valid uploadId.
    .partNumber(1) // Ensure the part number is correct.
    .copySourceRange("bytes=0-1023") // Adjust range as needed
    .build();

        return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
    })
    .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
    .whenComplete((uploadPartCopyResponse, exception) -> {
        if (exception != null) {
            // Handle any exceptions.
            logger.error("Error during upload part copy: " +
exception.getMessage());
        } else {
            // Successfully completed the upload part copy.
            System.out.println("Upload Part Copy completed successfully.
ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
        }
    });
    return null;
}

private static ByteBuffer getRandomByteBuffer(int size) {
    ByteBuffer buffer = ByteBuffer.allocate(size);
    for (int i = 0; i < size; i++) {
        buffer.put((byte) (Math.random() * 256));
    }
    buffer.flip();
    return buffer;
}
}
}

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK for Java 2.x .

- [CopyObject](#)

- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Innanzitutto, importa tutti i moduli necessari.

```
// Used to check if currently running file is this file.
import { fileURLToPath } from "node:url";
import { readdirSync, readFileSync, writeFileSync } from "node:fs";

// Local helper utils.
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";
import { wrapText } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

import {
  S3Client,
  CreateBucketCommand,
  PutObjectCommand,
  ListObjectsCommand,
  CopyObjectCommand,
  GetObjectCommand,
  DeleteObjectsCommand,
  DeleteBucketCommand,
} from "@aws-sdk/client-s3";
```

Le importazioni precedenti fanno riferimento ad alcune utilità di supporto. Queste utilità sono locali al GitHub repository collegato all'inizio di questa sezione. Come riferimento, consulta le implementazioni di tali utilità riportate di seguito.

```
export const dirnameFromMetaUrl = (metaUrl) =>
  fileURLToPath(new URL(".", metaUrl));

import { select, input, confirm, checkbox, password } from "@inquirer/prompts";

export class Prompter {
  /**
   * @param {{ message: string, choices: { name: string, value: string }[] }}
   options
   */
  select(options) {
    return select(options);
  }

  /**
   * @param {{ message: string }} options
   */
  input(options) {
    return input(options);
  }

  /**
   * @param {{ message: string }} options
   */
  password(options) {
    return password({ ...options, mask: true });
  }

  /**
   * @param {string} prompt
   */
  checkContinue = async (prompt = "") => {
    const prefix = prompt && `${prompt} `;
    const ok = await this.confirm({
      message: `${prefix}Continue?`,
    });
    if (!ok) throw new Error("Exiting...");
  };
}
```

```

/**
 * @param {{ message: string }} options
 */
confirm(options) {
  return confirm(options);
}

/**
 * @param {{ message: string, choices: { name: string, value: string }[] }}
options
 */
checkbox(options) {
  return checkbox(options);
}
}

export const wrapText = (text, char = "=") => {
  const rule = char.repeat(80);
  return `${rule}\n  ${text}\n${rule}\n`;
};

```

Gli oggetti vengono archiviati in 'bucket'. Definiamo una funzione per creare un nuovo bucket.

```

export const createBucket = async () => {
  const bucketName = await prompter.input({
    message: "Enter a bucket name. Bucket names must be globally unique:",
  });
  const command = new CreateBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log("Bucket created successfully.\n");
  return bucketName;
};

```

I bucket contengono "oggetti". Questa funzione carica il contenuto di una directory nel bucket come oggetti.

```

export const uploadFilesToBucket = async ({ bucketName, folderPath }) => {
  console.log(`Uploading files from ${folderPath}\n`);
  const keys = readdirSync(folderPath);
  const files = keys.map((key) => {
    const filePath = `${folderPath}/${key}`;

```

```
const fileContent = readFileSync(filePath);
return {
  Key: key,
  Body: fileContent,
};
});

for (const file of files) {
  await s3Client.send(
    new PutObjectCommand({
      Bucket: bucketName,
      Body: file.Body,
      Key: file.Key,
    }),
  );
  console.log(`${file.Key} uploaded successfully.`);
}
};
```

Dopo aver caricato gli oggetti, verifica che siano stati caricati correttamente. Puoi usare `ListObjects` per questo. Utilizzerai la proprietà `'Key'`, ma ci sono anche altre proprietà utili nella risposta.

```
export const listFilesInBucket = async ({ bucketName }) => {
  const command = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(command);
  const contentsList = Contents.map((c) => ` • ${c.Key}`).join("\n");
  console.log("\nHere's a list of files in the bucket:");
  console.log(`${contentsList}\n`);
};
```

A volte potresti voler copiare un oggetto da un bucket ad altri bucket. Usa il `CopyObject` comando per questo.

```
export const copyFileFromBucket = async ({ destinationBucket }) => {
  const proceed = await prompter.confirm({
    message: "Would you like to copy an object from another bucket?",
  });

  if (!proceed) {
```

```
    return;
  }
  const copy = async () => {
    try {
      const sourceBucket = await prompter.input({
        message: "Enter source bucket name:",
      });
      const sourceKey = await prompter.input({
        message: "Enter source key:",
      });
      const destinationKey = await prompter.input({
        message: "Enter destination key:",
      });

      const command = new CopyObjectCommand({
        Bucket: destinationBucket,
        CopySource: `${sourceBucket}/${sourceKey}`,
        Key: destinationKey,
      });
      await s3Client.send(command);
      await copyFileFromBucket({ destinationBucket });
    } catch (err) {
      console.error("Copy error.");
      console.error(err);
      const retryAnswer = await prompter.confirm({ message: "Try again?" });
      if (retryAnswer) {
        await copy();
      }
    }
  };
  await copy();
};
```

Non esiste un metodo SDK per ottenere più oggetti da un bucket. Creerai invece un elenco di oggetti da scaricare ed eseguirai iterazioni su di essi.

```
export const downloadFilesFromBucket = async ({ bucketName }) => {
  const { Contents } = await s3Client.send(
    new ListObjectsCommand({ Bucket: bucketName }),
  );
  const path = await prompter.input({
    message: "Enter destination path for files:",
  });
```

```
});

for (const content of Contents) {
  const obj = await s3Client.send(
    new GetObjectCommand({ Bucket: bucketName, Key: content.Key }),
  );
  writeFileSync(
    `${path}/${content.Key}`,
    await obj.Body.transformToByteArray(),
  );
}
console.log("Files downloaded successfully.\n");
};
```

È il momento di eseguire una pulizia delle risorse. Prima di poter essere eliminato, un bucket deve essere vuoto. Queste due funzioni svuotano ed eliminano il bucket.

```
export const emptyBucket = async ({ bucketName }) => {
  const listObjectsCommand = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(listObjectsCommand);
  const keys = Contents.map((c) => c.Key);

  const deleteObjectsCommand = new DeleteObjectsCommand({
    Bucket: bucketName,
    Delete: { Objects: keys.map((key) => ({ Key: key })) },
  });
  await s3Client.send(deleteObjectsCommand);
  console.log(`${bucketName} emptied successfully.\n`);
};

export const deleteBucket = async ({ bucketName }) => {
  const command = new DeleteBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log(`${bucketName} deleted successfully.\n`);
};
```

La funzione "principale" esegue entrambe le operazioni. Se esegui direttamente questo file, verrà chiamata la funzione principale.

```
const main = async () => {
```

```
const OBJECT_DIRECTORY = `${dirnameFromMetaUrl(
  import.meta.url,
)}../../../../resources/sample_files/.sample_media`;

try {
  console.log(wrapText("Welcome to the Amazon S3 getting started example.));
  console.log("Let's create a bucket.");
  const bucketName = await createBucket();
  await prompter.confirm({ message: continueMessage });

  console.log(wrapText("File upload.));
  console.log(
    "I have some default files ready to go. You can edit the source code to
provide your own.",
  );
  await uploadFilesToBucket({
    bucketName,
    folderPath: OBJECT_DIRECTORY,
  });

  await listFilesInBucket({ bucketName });
  await prompter.confirm({ message: continueMessage });

  console.log(wrapText("Copy files.));
  await copyFileFromBucket({ destinationBucket: bucketName });
  await listFilesInBucket({ bucketName });
  await prompter.confirm({ message: continueMessage });

  console.log(wrapText("Download files.));
  await downloadFilesFromBucket({ bucketName });

  console.log(wrapText("Clean up.));
  await emptyBucket({ bucketName });
  await deleteBucket({ bucketName });
} catch (err) {
  console.error(err);
}
};
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per JavaScript .
 - [CopyObject](#)

- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Kotlin

SDK per Kotlin

Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun main(args: Array<String>) {  
    val usage = ""  
    Usage:  
        <bucketName> <key> <objectPath> <savePath> <toBucket>  
  
    Where:  
        bucketName - The Amazon S3 bucket to create.  
        key - The key to use.  
        objectPath - The path where the file is located (for example, C:/AWS/  
book2.pdf).  
        savePath - The path where the file is saved after it's downloaded (for  
example, C:/AWS/book2.pdf).  
        toBucket - An Amazon S3 bucket to where an object is copied to (for  
example, C:/AWS/book2.pdf).  
    ""  
  
    if (args.size != 4) {  
        println(usage)  
        exitProcess(1)  
    }  
  
    val bucketName = args[0]
```



```
val key = args[1]
val objectPath = args[2]
val savePath = args[3]
val toBucket = args[4]

// Create an Amazon S3 bucket.
createBucket(bucketName)

// Update a local file to the Amazon S3 bucket.
putObject(bucketName, key, objectPath)

// Download the object to another local file.
getObjectFromMrap(bucketName, key, savePath)

// List all objects located in the Amazon S3 bucket.
listBucketObs(bucketName)

// Copy the object to another Amazon S3 bucket
copyBucketOb(bucketName, key, toBucket)

// Delete the object from the Amazon S3 bucket.
deleteBucketObs(bucketName, key)

// Delete the Amazon S3 bucket.
deleteBucket(bucketName)
println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(
    bucketName: String,
    objectKey: String,
    objectPath: String,
```

```
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            this.body = Paths.get(objectPath).asByteStream()
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}

suspend fun getObjectFromMrap(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}

suspend fun listBucketObs(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }
}
```

```
S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->

    val response = s3.listObjects(request)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}

suspend fun copyBucketOb(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedOperationException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
```

```
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request =
        DeleteBucketRequest {
            bucket = bucketName
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Kotlin.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

PHP

SDK per PHP

 Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
echo("\n");
echo("-----\n");
print("Welcome to the Amazon S3 getting started demo using PHP!\n");
echo("-----\n");

$region = 'us-west-2';

$this->s3client = new S3Client([
    'region' => $region,
]);
/* Inline declaration example
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
*/

$this->bucketName = "amzn-s3-demo-bucket-" . uniqid();

try {
    $this->s3client->createBucket([
        'Bucket' => $this->bucketName,
        'CreateBucketConfiguration' => ['LocationConstraint' => $region],
    ]);
    echo "Created bucket named: $this->bucketName \n";
} catch (Exception $exception) {
    echo "Failed to create bucket $this->bucketName with error: " .
    $exception->getMessage();
    exit("Please fix error with bucket creation before continuing.");
}

$fileName = __DIR__ . "/local-file-" . uniqid();
try {
    $this->s3client->putObject([
        'Bucket' => $this->bucketName,
```

```
        'Key' => $fileName,
        'SourceFile' => __DIR__ . '/testfile.txt'
    ]);
    echo "Uploaded $fileName to $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
    exit("Please fix error with file upload before continuing.");
}

try {
    $file = $this->s3client->getObject([
        'Bucket' => $this->bucketName,
        'Key' => $fileName,
    ]);
    $body = $file->get('Body');
    $body->rewind();
    echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
} catch (Exception $exception) {
    echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
    exit("Please fix error with file downloading before continuing.");
}

try {
    $folder = "copied-folder";
    $this->s3client->copyObject([
        'Bucket' => $this->bucketName,
        'CopySource' => "$this->bucketName/$fileName",
        'Key' => "$folder/$fileName-copy",
    ]);
    echo "Copied $fileName to $folder/$fileName-copy.\n";
} catch (Exception $exception) {
    echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
    exit("Please fix error with object copying before continuing.");
}

try {
    $contents = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    echo "The contents of your bucket are: \n";
    foreach ($contents['Contents'] as $content) {
```

```
        echo $content['Key'] . "\n";
    }
} catch (Exception $exception) {
    echo "Failed to list objects in $this->bucketName with error: " .
$exception->getMessage();
    exit("Please fix error with listing objects before continuing.");
}

try {
    $objects = [];
    foreach ($contents['Contents'] as $content) {
        $objects[] = [
            'Key' => $content['Key'],
        ];
    }
    $this->s3client->deleteObjects([
        'Bucket' => $this->bucketName,
        'Delete' => [
            'Objects' => $objects,
        ],
    ]);
    $check = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    if (isset($check['Contents']) && count($check['Contents']) > 0) {
        throw new Exception("Bucket wasn't empty.");
    }
    echo "Deleted all objects and folders from $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to delete $fileName from $this->bucketName with error:
" . $exception->getMessage();
    exit("Please fix error with object deletion before continuing.");
}

try {
    $this->s3client->deleteBucket([
        'Bucket' => $this->bucketName,
    ]);
    echo "Deleted bucket $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
    exit("Please fix error with bucket deletion before continuing.");
}
```

```
echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per PHP .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Python

SDK per Python (Boto3)

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import io
import os
import uuid

import boto3
from boto3.s3.transfer import S3UploadFailedError
from botocore.exceptions import ClientError

def do_scenario(s3_resource):
    print("-" * 88)
    print("Welcome to the Amazon S3 getting started demo!")
    print("-" * 88)
```



```

bucket_name = f"amzn-s3-demo-bucket-{uuid.uuid4()}"
bucket = s3_resource.Bucket(bucket_name)
try:
    bucket.create(
        CreateBucketConfiguration={
            "LocationConstraint": s3_resource.meta.client.meta.region_name
        }
    )
    print(f"Created demo bucket named {bucket.name}.")
except ClientError as err:
    print(f"Tried and failed to create demo bucket {bucket_name}.")
    print(f"\t{err.response['Error']['Code']}: {err.response['Error']
['Message']}")
    print(f"\nCan't continue the demo without a bucket!")
    return

file_name = None
while file_name is None:
    file_name = input("\nEnter a file you want to upload to your bucket: ")
    if not os.path.exists(file_name):
        print(f"Couldn't find file {file_name}. Are you sure it exists?")
        file_name = None

obj = bucket.Object(os.path.basename(file_name))
try:
    obj.upload_file(file_name)
    print(
        f"Uploaded file {file_name} into bucket {bucket.name} with key
{obj.key}."
    )
except S3UploadFailedError as err:
    print(f"Couldn't upload file {file_name} to {bucket.name}.")
    print(f"\t{err}")

answer = input(f"\nDo you want to download {obj.key} into memory (y/n)? ")
if answer.lower() == "y":
    data = io.BytesIO()
    try:
        obj.download_fileobj(data)
        data.seek(0)
        print(f"Got your object. Here are the first 20 bytes:\n")
        print(f"\t{data.read(20)}")
    except ClientError as err:

```

```

        print(f"Couldn't download {obj.key}.")
        print(
            f"\t{err.response['Error']['Code']}: {err.response['Error']
['Message']}"
        )

    answer = input(
        f"\nDo you want to copy {obj.key} to a subfolder in your bucket (y/n)? "
    )
    if answer.lower() == "y":
        dest_obj = bucket.Object(f"demo-folder/{obj.key}")
        try:
            dest_obj.copy({"Bucket": bucket.name, "Key": obj.key})
            print(f"Copied {obj.key} to {dest_obj.key}.")
        except ClientError as err:
            print(f"Couldn't copy {obj.key} to {dest_obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}: {err.response['Error']
['Message']}"
            )

    print("\nYour bucket contains the following objects:")
    try:
        for o in bucket.objects.all():
            print(f"\t{o.key}")
    except ClientError as err:
        print(f"Couldn't list the objects in bucket {bucket.name}.")
        print(f"\t{err.response['Error']['Code']}: {err.response['Error']
['Message']}"
    )

    answer = input(
        "\nDo you want to delete all of the objects as well as the bucket (y/n)?
"
    )
    if answer.lower() == "y":
        try:
            bucket.objects.delete()
            bucket.delete()
            print(f"Emptied and deleted bucket {bucket.name}.\n")
        except ClientError as err:
            print(f"Couldn't empty and delete bucket {bucket.name}.")
            print(
                f"\t{err.response['Error']['Code']}: {err.response['Error']
['Message']}"
            )

```

```
)

print("Thanks for watching!")
print("-" * 88)

if __name__ == "__main__":
    do_scenario(boto3.resource("s3"))
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Python (Boto3).
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Ruby

SDK per Ruby

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
require 'aws-sdk-s3'

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
```

```
@s3_resource = s3_resource
end

# Creates a bucket with a random name in the currently configured account and
# AWS Region.
#
# @return [Aws::S3::Bucket] The newly created bucket.
def create_bucket
  bucket = @s3_resource.create_bucket(
    bucket: "amzn-s3-demo-bucket-#{Random.uuid}",
    create_bucket_configuration: {
      location_constraint: 'us-east-1' # NOTE: only certain regions permitted
    }
  )
  puts("Created demo bucket named #{bucket.name}.")
rescue Aws::Errors::ServiceError => e
  puts('Tried and failed to create demo bucket.')
  puts("\t#{e.code}: #{e.message}")
  puts("\nCan't continue the demo without a bucket!")
  raise
else
  bucket
end

# Requests a file name from the user.
#
# @return The name of the file.
def create_file
  File.open('demo.txt', w) { |f| f.write('This is a demo file.') }
end

# Uploads a file to an Amazon S3 bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket object representing the upload
destination
# @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded
file.
def upload_file(bucket)
  File.open('demo.txt', 'w+') { |f| f.write('This is a demo file.') }
  s3_object = bucket.object(File.basename('demo.txt'))
  s3_object.upload_file('demo.txt')
  puts("Uploaded file demo.txt into bucket #{bucket.name} with key
#{s3_object.key}.")
rescue Aws::Errors::ServiceError => e
```

```
puts("Couldn't upload file demo.txt to #{bucket.name}.")
puts("\t#{e.code}: #{e.message}")
raise
else
  s3_object
end

# Downloads an Amazon S3 object to a file.
#
# @param s3_object [Aws::S3::Object] The object to download.
def download_file(s3_object)
  puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
  answer = gets.chomp.downcase
  if answer == 'y'
    puts('Enter a name for the downloaded file: ')
    file_name = gets.chomp
    s3_object.download_file(file_name)
    puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't download #{s3_object.key}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

# Copies an Amazon S3 object to a subfolder within the same bucket.
#
# @param source_object [Aws::S3::Object] The source object to copy.
# @return [Aws::S3::Object, nil] The destination object.
def copy_object(source_object)
  dest_object = nil
  puts("\nDo you want to copy #{source_object.key} to a subfolder in your
bucket (y/n)? ")
  answer = gets.chomp.downcase
  if answer == 'y'
    dest_object = source_object.bucket.object("demo-folder/
#{source_object.key}")
    dest_object.copy_from(source_object)
    puts("Copied #{source_object.key} to #{dest_object.key}.")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't copy #{source_object.key}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

```
else
  dest_object
end

# Lists the objects in an Amazon S3 bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to query.
def list_objects(bucket)
  puts("\nYour bucket contains the following objects:")
  bucket.objects.each do |obj|
    puts("\t#{obj.key}")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't list the objects in bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?")
  answer = gets.chomp.downcase
  if answer == 'y'
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

# Runs the Amazon S3 getting started scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the Amazon S3 getting started demo!')
  puts('-' * 88)

  bucket = scenario.create_bucket
```

```
s3_object = scenario.upload_file(bucket)
scenario.download_file(s3_object)
scenario.copy_object(s3_object)
scenario.list_objects(bucket)
scenario.delete_bucket(bucket)

puts('Thanks for watching!')
puts('-' * 88)
rescue Aws::Errors::ServiceError
  puts('Something went wrong with the demo!')
end

run_scenario(ScenarioGettingStarted.new(Aws::S3::Resource.new)) if $PROGRAM_NAME
== __FILE__
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Ruby .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Codice per la crate binario che esegue lo scenario.

```

#![allow(clippy::result_large_err)]

//! Purpose
//! Shows how to use the AWS SDK for Rust to get started using
//! Amazon Simple Storage Service (Amazon S3). Create a bucket, move objects
    into and out of it,
//! and delete all resources at the end of the demo.
//!
//! This example follows the steps in "Getting started with Amazon S3" in the
    Amazon S3
//! user guide.
//! - https://docs.aws.amazon.com/AmazonS3/latest/userguide/
    GetStartedWithS3.html

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::{config::Region, Client};
use s3_code_examples::error::S3ExampleError;
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), S3ExampleError> {
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();
    let shared_config =
aws_config::from_env().region(region_provider).load().await;
    let client = Client::new(&shared_config);
    let bucket_name = format!("amzn-s3-demo-bucket-{}", Uuid::new_v4());
    let file_name = "s3/testfile.txt".to_string();
    let key = "test file key name".to_string();
    let target_key = "target_key".to_string();

    if let Err(e) = run_s3_operations(region, client, bucket_name, file_name,
key, target_key).await
    {
        eprintln!("{:?}", e);
    };

    Ok(())
}

async fn run_s3_operations(
    region: Region,
    client: Client,

```



```

    bucket_name: String,
    file_name: String,
    key: String,
    target_key: String,
) -> Result<(), S3ExampleError> {
    s3_code_examples::create_bucket(&client, &bucket_name, &region).await?;
    let run_example: Result<(), S3ExampleError> = (async {
        s3_code_examples::upload_object(&client, &bucket_name, &file_name,
&key).await?;
        let _object = s3_code_examples::download_object(&client, &bucket_name,
&key).await;
        s3_code_examples::copy_object(&client, &bucket_name, &bucket_name, &key,
&target_key)
            .await?;
        s3_code_examples::list_objects(&client, &bucket_name).await?;
        s3_code_examples::clear_bucket(&client, &bucket_name).await?;
        Ok(())
    })
    .await;
    if let Err(err) = run_example {
        eprintln!("Failed to complete getting-started example: {err:?}");
    }
    s3_code_examples::delete_bucket(&client, &bucket_name).await?;

    Ok(())
}

```

Azioni comuni utilizzate dallo scenario.

```

pub async fn create_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    region: &aws_config::Region,
) -> Result<Option<aws_sdk_s3::operation::create_bucket::CreateBucketOutput>,
S3ExampleError> {
    let constraint =
aws_sdk_s3::types::BucketLocationConstraint::from(region.to_string().as_str());
    let cfg = aws_sdk_s3::types::CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    let create = client
        .create_bucket()

```

```

        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await;

    // BucketAlreadyExists and BucketAlreadyOwnedByYou are not problems for this
    task.
    create.map(Some).or_else(|err| {
        if err
            .as_service_error()
            .map(|se| se.is_bucket_already_exists() ||
se.is_bucket_already_owned_by_you())
            == Some(true)
        {
            Ok(None)
        } else {
            Err(S3ExampleError::from(err))
        }
    })
}

pub async fn upload_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::put_object::PutObjectOutput, S3ExampleError> {
    let body =
aws_sdk_s3::primitives::ByteStream::from_path(std::path::Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
        .map_err(S3ExampleError::from)
}

pub async fn download_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::get_object::GetObjectOutput, S3ExampleError> {

```

```

        client
            .get_object()
            .bucket(bucket_name)
            .key(key)
            .send()
            .await
            .map_err(S3ExampleError::from)
    }

    /// Copy an object from one bucket to another.
    pub async fn copy_object(
        client: &aws_sdk_s3::Client,
        source_bucket: &str,
        destination_bucket: &str,
        source_object: &str,
        destination_object: &str,
    ) -> Result<(), S3ExampleError> {
        let source_key = format!("{source_bucket}/{source_object}");
        let response = client
            .copy_object()
            .copy_source(&source_key)
            .bucket(destination_bucket)
            .key(destination_object)
            .send()
            .await?;

        println!(
            "Copied from {source_key} to {destination_bucket}/{destination_object}
            with etag {}",
            response
                .copy_object_result
                .unwrap_or_else(||
                    aws_sdk_s3::types::CopyObjectResult::builder().build())
                .e_tag()
                .unwrap_or("missing")
        );
        Ok(())
    }

    pub async fn list_objects(client: &aws_sdk_s3::Client, bucket: &str) ->
    Result<(), S3ExampleError> {
        let mut response = client
            .list_objects_v2()
            .bucket(bucket.to_owned())

```

```

        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

while let Some(result) = response.next().await {
    match result {
        Ok(output) => {
            for object in output.contents() {
                println!(" - {}", object.key().unwrap_or("Unknown"));
            }
        }
        Err(err) => {
            eprintln!("{err:?}")
        }
    }
}

Ok(())
}

/// Given a bucket, remove all objects in the bucket, and then ensure no objects
/// remain in the bucket.
pub async fn clear_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<Vec<String>, S3ExampleError> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    // delete_objects no longer needs to be mutable.
    let objects_to_delete: Vec<String> = objects
        .contents()
        .iter()
        .filter_map(|obj| obj.key())
        .map(String::from)
        .collect();

    if objects_to_delete.is_empty() {
        return Ok(vec![]);
    }

    let return_keys = objects_to_delete.clone();

    delete_objects(client, bucket_name, objects_to_delete).await?;
}

```

```

    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    eprintln!("{objects:?}");

    match objects.key_count {
        Some(0) => Ok(return_keys),
        _ => Err(S3ExampleError::new(
            "There were still objects left in the bucket.",
        )),
    }
}

pub async fn delete_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<(), S3ExampleError> {
    let resp = client.delete_bucket().bucket(bucket_name).send().await;
    match resp {
        Ok(_) => Ok(()),
        Err(err) => {
            if err
                .as_service_error()
                .and_then(aws_sdk_s3::error::ProvideErrorMetadata::code)
                == Some("NoSuchBucket")
            {
                Ok(())
            } else {
                Err(S3ExampleError::from(err))
            }
        }
    }
}
}

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API SDK AWS per Rust.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)

- [ListObjectsV2](#)
- [PutObject](#)

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

" Create an Amazon Simple Storage Service (Amazon S3) bucket. "
TRY.
    " determine our region from our session
    DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
    DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
    " When in the us-east-1 region, you must not specify a constraint
    " In all other regions, specify the region as the constraint
    IF lv_region = 'us-east-1'.
        CLEAR lo_constraint.
    ELSE.
        lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
    ENDIF.

    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
        io_createbucketconfiguration = lo_constraint ).
    MESSAGE 'S3 bucket created.' TYPE 'I'.
CATCH /aws1/cx_s3_bucketalrddyexists.
    MESSAGE 'Bucket name already exists.' TYPE 'E'.
CATCH /aws1/cx_s3_bktalrddyownedbyyou.
    MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
ENDTRY.
```

```
"Upload an object to an S3 bucket."
TRY.
    "Get contents of file from application server."
    DATA lv_file_content TYPE xstring.
    OPEN DATASET iv_key FOR INPUT IN BINARY MODE.
    READ DATASET iv_key INTO lv_file_content.
    CLOSE DATASET iv_key.

    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key
        iv_body = lv_file_content ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" Get an object from a bucket. "
TRY.
    DATA(lo_result) = lo_s3->getobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key ).
    DATA(lv_object_data) = lo_result->get_body( ).
    MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

" Copy an object to a subfolder in a bucket. "
TRY.
    lo_s3->copyobject(
        iv_bucket = iv_bucket_name
        iv_key = |{ iv_copy_to_folder }/{ iv_key }|
        iv_copysource = |{ iv_bucket_name }/{ iv_key }| ).
    MESSAGE 'Object copied to a subfolder.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

" List objects in the bucket. "
```

```

TRY.
    DATA(lo_list) = lo_s3->listobjects(
        iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

DATA text TYPE string VALUE 'Object List - '.
DATA lv_object_key TYPE /aws1/s3_objectkey.
LOOP AT lo_list->get_contents( ) INTO DATA(lo_object).
    lv_object_key = lo_object->get_key( ).
    CONCATENATE lv_object_key ', ' INTO text.
ENDLOOP.
MESSAGE text TYPE 'I'.

" Delete the objects in a bucket. "
TRY.
    lo_s3->deleteobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key ).
    lo_s3->deleteobject(
        iv_bucket = iv_bucket_name
        iv_key = |{ iv_copy_to_folder }/{ iv_key }| ).
    MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" Delete the bucket. "
TRY.
    lo_s3->deletebucket(
        iv_bucket = iv_bucket_name ).
    MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- Per informazioni dettagliate sull'API, consulta gli argomenti seguenti nella documentazione di riferimento dell'API AWS SDK per SAP ABAP.
- [CopyObject](#)

- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import AWSS3

import Foundation
import AWSS3
import Smithy
import ClientRuntime

/// A class containing all the code that interacts with the AWS SDK for Swift.
public class ServiceHandler {
    let configuration: S3Client.S3ClientConfiguration
    let client: S3Client

    enum HandlerError: Error {
        case getObjectBody(String)
        case readGetObjectBody(String)
        case missingContents(String)
    }

    /// Initialize and return a new ``ServiceHandler`` object, which is used to
    drive the AWS calls
    /// used for the example.
    ///
```

```
/// - Returns: A new ``ServiceHandler`` object, ready to be called to
///           execute AWS operations.
public init() async throws {
    do {
        configuration = try await S3Client.S3ClientConfiguration()
        // configuration.region = "us-east-2" // Uncomment this to set the
region programmatically.
        client = S3Client(config: configuration)
    }
    catch {
        print("ERROR: ", dump(error, name: "Initializing S3 client"))
        throw error
    }
}

/// Create a new user given the specified name.
///
/// - Parameters:
///   - name: Name of the bucket to create.
/// Throws an exception if an error occurs.
public func createBucket(name: String) async throws {
    var input = CreateBucketInput(
        bucket: name

        // For regions other than "us-east-1", you must set the
locationConstraint in the createBucketConfiguration.
        // For more information, see LocationConstraint in the S3 API guide.
        // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_CreateBucket.html#API_CreateBucket_RequestBody
        if let region = configuration.region {
            if region != "us-east-1" {
                input.createBucketConfiguration =
S3ClientTypes.CreateBucketConfiguration(locationConstraint:
S3ClientTypes.BucketLocationConstraint(rawValue: region))
            }
        }

        do {
            _ = try await client.createBucket(input: input)
        }
        catch let error as BucketAlreadyOwnedByYou {
```

```
        print("The bucket '\(name)' already exists and is owned by you. You
may wish to ignore this exception.")
        throw error
    }
    catch {
        print("ERROR: ", dump(error, name: "Creating a bucket"))
        throw error
    }
}

/// Delete a bucket.
/// - Parameter name: Name of the bucket to delete.
public func deleteBucket(name: String) async throws {
    let input = DeleteBucketInput(
        bucket: name
    )
    do {
        _ = try await client.deleteBucket(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Deleting a bucket"))
        throw error
    }
}

/// Upload a file from local storage to the bucket.
/// - Parameters:
///   - bucket: Name of the bucket to upload the file to.
///   - key: Name of the file to create.
///   - file: Path name of the file to upload.
public func uploadFile(bucket: String, key: String, file: String) async
throws {
    let fileUrl = URL(fileURLWithPath: file)
    do {
        let fileData = try Data(contentsOf: fileUrl)
        let dataStream = ByteStream.data(fileData)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )

        _ = try await client.putObject(input: input)
```

```

    }
    catch {
        print("ERROR: ", dump(error, name: "Putting an object. "))
        throw error
    }
}

/// Create a file in the specified bucket with the given name. The new
/// file's contents are uploaded from a `Data` object.
///
/// - Parameters:
///   - bucket: Name of the bucket to create a file in.
///   - key: Name of the file to create.
///   - data: A `Data` object to write into the new file.
public func createFile(bucket: String, key: String, withData data: Data)
async throws {
    let dataStream = ByteStream.data(data)

    let input = PutObjectInput(
        body: dataStream,
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.putObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Putting an object. "))
        throw error
    }
}

/// Download the named file to the given directory on the local device.
///
/// - Parameters:
///   - bucket: Name of the bucket that contains the file to be copied.
///   - key: The name of the file to copy from the bucket.
///   - to: The path of the directory on the local device where you want to
///         download the file.
public func downloadFile(bucket: String, key: String, to: String) async
throws {
    let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

```

```
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }

            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
            }

            try data.write(to: fileUrl)
        }
        catch {
            print("ERROR: ", dump(error, name: "Downloading a file. "))
            throw error
        }
    }
}

/// Read the specified file from the given S3 bucket into a Swift
/// `Data` object.
///
/// - Parameters:
///   - bucket: Name of the bucket containing the file to read.
///   - key: Name of the file within the bucket to read.
///
/// - Returns: A `Data` object containing the complete file data.
public func readFile(bucket: String, key: String) async throws -> Data {
    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )
    do {
        let output = try await client.getObject(input: input)

        guard let body = output.body else {
            throw HandlerError.getObjectBody("GetObjectInput missing body.")
        }
    }
```

```
        guard let data = try await body.readData() else {
            throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
        }

        return data
    }
    catch {
        print("ERROR: ", dump(error, name: "Reading a file. "))
        throw error
    }
}

/// Copy a file from one bucket to another.
///
/// - Parameters:
///   - sourceBucket: Name of the bucket containing the source file.
///   - name: Name of the source file.
///   - destBucket: Name of the bucket to copy the file into.
public func copyFile(from sourceBucket: String, name: String, to destBucket:
String) async throws {
    let srcUrl = ("\(sourceBucket)/
\((name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

    let input = CopyObjectInput(
        bucket: destBucket,
        copySource: srcUrl,
        key: name
    )
    do {
        _ = try await client.copyObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Copying an object. "))
        throw error
    }
}

/// Deletes the specified file from Amazon S3.
///
/// - Parameters:
///   - bucket: Name of the bucket containing the file to delete.
///   - key: Name of the file to delete.
```

```
///
public func deleteFile(bucket: String, key: String) async throws {
    let input = DeleteObjectInput(
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.deleteObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Deleting a file. "))
        throw error
    }
}

/// Returns an array of strings, each naming one file in the
/// specified bucket.
///
/// - Parameter bucket: Name of the bucket to get a file listing for.
/// - Returns: An array of `String` objects, each giving the name of
///            one file contained in the bucket.
public func listBucketFiles(bucket: String) async throws -> [String] {
    do {
        let input = ListObjectsV2Input(
            bucket: bucket
        )

        // Use "Paginated" to get all the objects.
        // This lets the SDK handle the 'continuationToken' in
        "ListObjectsV2Output".
        let output = client.listObjectsV2Paginated(input: input)
        var names: [String] = []

        for try await page in output {
            guard let objList = page.contents else {
                print("ERROR: listObjectsV2Paginated returned nil contents.")
                continue
            }

            for obj in objList {
                if let objName = obj.key {
                    names.append(objName)
                }
            }
        }
    }
}
```

```

        }
    }

    return names
}
catch {
    print("ERROR: ", dump(error, name: "Listing objects. "))
    throw error
}
}
}

```

```

import AWSS3

import Foundation
import ServiceHandler
import ArgumentParser

/// The command-line arguments and options available for this
/// example command.
struct ExampleCommand: ParsableCommand {
    @Argument(help: "Name of the S3 bucket to create")
    var bucketName: String

    @Argument(help: "Pathname of the file to upload to the S3 bucket")
    var uploadSource: String

    @Argument(help: "The name (key) to give the file in the S3 bucket")
    var objName: String

    @Argument(help: "S3 bucket to copy the object to")
    var destBucket: String

    @Argument(help: "Directory where you want to download the file from the S3
bucket")
    var downloadDir: String

    static var configuration = CommandConfiguration(
        commandName: "s3-basics",
        abstract: "Demonstrates a series of basic AWS S3 functions.",
        discussion: ""
    )
}

```



```
    Performs the following Amazon S3 commands:

    * `CreateBucket`
    * `PutObject`
    * `GetObject`
    * `CopyObject`
    * `ListObjects`
    * `DeleteObjects`
    * `DeleteBucket`
    ""
)

/// Called by ``main()`` to do the actual running of the AWS
/// example.
func runAsync() async throws {
    let serviceHandler = try await ServiceHandler()

    // 1. Create the bucket.
    print("Creating the bucket \(bucketName)...")
    try await serviceHandler.createBucket(name: bucketName)

    // 2. Upload a file to the bucket.
    print("Uploading the file \(uploadSource)...")
    try await serviceHandler.uploadFile(bucket: bucketName, key: objName,
file: uploadSource)

    // 3. Download the file.
    print("Downloading the file \(objName) to \(downloadDir)...")
    try await serviceHandler.downloadFile(bucket: bucketName, key: objName,
to: downloadDir)

    // 4. Copy the file to another bucket.
    print("Copying the file to the bucket \(destBucket)...")
    try await serviceHandler.copyFile(from: bucketName, name: objName, to:
destBucket)

    // 5. List the contents of the bucket.

    print("Getting a list of the files in the bucket \(bucketName)")
    let fileList = try await serviceHandler.listBucketFiles(bucket:
bucketName)
    let numFiles = fileList.count
    if numFiles != 0 {
```

```
        print("\(numFiles) file\((numFiles > 1) ? "s" : "") in bucket
        \(bucketName):")
        for name in fileList {
            print("  \(name)")
        }
    } else {
        print("No files found in bucket \(bucketName)")
    }

    // 6. Delete the objects from the bucket.

    print("Deleting the file \(objName) from the bucket \(bucketName)...")
    try await serviceHandler.deleteFile(bucket: bucketName, key: objName)
    print("Deleting the file \(objName) from the bucket \(destBucket)...")
    try await serviceHandler.deleteFile(bucket: destBucket, key: objName)

    // 7. Delete the bucket.
    print("Deleting the bucket \(bucketName)...")
    try await serviceHandler.deleteBucket(name: bucketName)

    print("Done.")
}
}

//
// Main program entry point.
//
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API SDK AWS per Swift.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Azioni per l'utilizzo di Amazon S3 AWS SDK

I seguenti esempi di codice mostrano come eseguire singole azioni Amazon S3 con AWS gli SDK. Ogni esempio include un collegamento a GitHub, dove puoi trovare le istruzioni per la configurazione e l'esecuzione del codice.

Questi estratti chiamano l'API Amazon S3 e sono estratti di codice di programmi più grandi che devono essere eseguiti in modo contestuale. È possibile visualizzare le azioni nel contesto in [Scenari per l'utilizzo di Amazon S3 AWS SDK](#).

Gli esempi seguenti includono solo le azioni più comunemente utilizzate. Per un elenco completo, consulta la [documentazione di riferimento dell'API Amazon Simple Storage](#).

Esempi

- [Da utilizzare AbortMultipartUploadcon un AWS SDK o CLI](#)
- [Utilizzare CompleteMultipartUploadcon un AWS SDK o CLI](#)
- [Utilizzare CopyObjectcon un AWS SDK o CLI](#)
- [Utilizzare CreateBucketcon un AWS SDK o CLI](#)
- [Utilizzare CreateMultiRegionAccessPointcon un AWS SDK](#)
- [Utilizzare CreateMultipartUploadcon un AWS SDK o CLI](#)
- [Utilizzare CreatePresignedPostcon un AWS SDK](#)

- [Utilizzare DeleteBucketcon un AWS SDK o CLI](#)
- [Utilizzare DeleteBucketAnalyticsConfiguration con una CLI](#)
- [Utilizzare DeleteBucketCorscon un AWS SDK o CLI](#)
- [Utilizzare DeleteBucketEncryption con una CLI](#)
- [Utilizzare DeleteBucketInventoryConfiguration con una CLI](#)
- [Utilizzare DeleteBucketLifecyclecon un AWS SDK o CLI](#)
- [Utilizzare DeleteBucketMetricsConfiguration con una CLI](#)
- [Utilizzare DeleteBucketPolicycon un AWS SDK o CLI](#)
- [Utilizzare DeleteBucketReplication con una CLI](#)
- [Utilizzare DeleteBucketTagging con una CLI](#)
- [Utilizzare DeleteBucketWebsitecon un AWS SDK o CLI](#)
- [Utilizzare DeleteObjectcon un AWS SDK o CLI](#)
- [Utilizzare DeleteObjectTagging con una CLI](#)
- [Utilizzare DeleteObjectscon un AWS SDK o CLI](#)
- [Utilizzare DeletePublicAccessBlock con una CLI](#)
- [Utilizzare GetBucketAccelerateConfiguration con una CLI](#)
- [Utilizzare GetBucketAclcon un AWS SDK o CLI](#)
- [Utilizzare GetBucketAnalyticsConfiguration con una CLI](#)
- [Utilizzare GetBucketCorscon un AWS SDK o CLI](#)
- [Utilizzare GetBucketEncryptioncon un AWS SDK o CLI](#)
- [Utilizzare GetBucketInventoryConfiguration con una CLI](#)
- [Utilizzare GetBucketLifecycleConfigurationcon un AWS SDK o CLI](#)
- [Utilizzare GetBucketLocationcon un AWS SDK o CLI](#)
- [Utilizzare GetBucketLogging con una CLI](#)
- [Utilizzare GetBucketMetricsConfiguration con una CLI](#)
- [Utilizzare GetBucketNotification con una CLI](#)
- [Utilizzare GetBucketPolicycon un AWS SDK o CLI](#)
- [Utilizzare GetBucketPolicyStatus con una CLI](#)
- [Utilizzare GetBucketReplicationcon un AWS SDK o CLI](#)
- [Utilizzare GetBucketRequestPayment con una CLI](#)

- [Utilizzare GetBucketTagging con una CLI](#)
- [Utilizzare GetBucketVersioning con una CLI](#)
- [Utilizzare GetBucketWebsite con un AWS SDK o CLI](#)
- [Utilizzare GetObject con un AWS SDK o CLI](#)
- [Utilizzare GetObjectAcl con un AWS SDK o CLI](#)
- [Utilizzare GetObjectAttributes con un AWS SDK o CLI](#)
- [Utilizzare GetObjectLegalHold con un AWS SDK o CLI](#)
- [Utilizzare GetObjectLockConfiguration con un AWS SDK o CLI](#)
- [Utilizzare GetObjectRetention con un AWS SDK o CLI](#)
- [Utilizzare GetObjectTagging con una CLI](#)
- [Utilizzare GetPublicAccessBlock con una CLI](#)
- [Utilizzare HeadBucket con un AWS SDK o CLI](#)
- [Utilizzare HeadObject con un AWS SDK o CLI](#)
- [Utilizzare ListBucketAnalyticsConfigurations con una CLI](#)
- [Utilizzare ListBucketInventoryConfigurations con una CLI](#)
- [Utilizzare ListBuckets con un AWS SDK o CLI](#)
- [Utilizzare ListMultipartUploads con un AWS SDK o CLI](#)
- [Utilizzare ListObjectVersions con un AWS SDK o CLI](#)
- [Utilizzare ListObjects con una CLI](#)
- [Usa ListObjectsV2 con un AWS SDK o CLI](#)
- [Utilizzare PutBucketAccelerateConfiguration con un AWS SDK o CLI](#)
- [Utilizzare PutBucketAcl con un AWS SDK o CLI](#)
- [Utilizzare PutBucketCors con un AWS SDK o CLI](#)
- [Utilizzare PutBucketEncryption con un AWS SDK o CLI](#)
- [Utilizzare PutBucketLifecycleConfiguration con un AWS SDK o CLI](#)
- [Utilizzare PutBucketLogging con un AWS SDK o CLI](#)
- [Utilizzare PutBucketNotification con una CLI](#)
- [Utilizzare PutBucketNotificationConfiguration con un AWS SDK o CLI](#)
- [Utilizzare PutBucketPolicy con un AWS SDK o CLI](#)
- [Utilizzare PutBucketReplication con un AWS SDK o CLI](#)

- [Utilizzare PutBucketRequestPayment con una CLI](#)
- [Utilizzare PutBucketTagging con una CLI](#)
- [Utilizzare PutBucketVersioning con un AWS SDK o CLI](#)
- [Utilizzare PutBucketWebsite con un AWS SDK o CLI](#)
- [Utilizzare PutObject con un AWS SDK o CLI](#)
- [Utilizzare PutObjectAcl con un AWS SDK o CLI](#)
- [Utilizzare PutObjectLegalHold con un AWS SDK o CLI](#)
- [Utilizzare PutObjectLockConfiguration con un AWS SDK o CLI](#)
- [Utilizzare PutObjectRetention con un AWS SDK o CLI](#)
- [Utilizzare RestoreObject con un AWS SDK o CLI](#)
- [Utilizzare SelectObjectContent con un AWS SDK o CLI](#)
- [Utilizzare UploadPart con un AWS SDK o CLI](#)
- [Utilizzare UploadPartCopy con un AWS SDK o CLI](#)

Da utilizzare **AbortMultipartUpload** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare **AbortMultipartUpload**.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- [Eliminare caricamenti in più parti incompleti](#)
- [Utilizzare la funzionalità per l'integrità degli oggetti di Amazon S3](#)

C++

SDK per C++

 Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
#!/ Abort a multipart upload to an S3 bucket.
```

```

/ * !
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                       const Aws::String &key,
                                       const Aws::String &uploadID,
                                       const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
        client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
        outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- Per i dettagli sull'API, consulta la [AbortMultipartUpload](#) sezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Come annullare il caricamento in più parti specificato

Il comando `abort-multipart-upload` seguente annulla un caricamento in più parti per la chiave `multipart/01` nel bucket `amzn-s3-demo-bucket`.

```
aws s3api abort-multipart-upload \  
  --bucket amzn-s3-demo-bucket \  
  --key multipart/01 \  
  --upload-  
id dfRtDYU0WMCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZLjF.Yxwh6XG7WfS2vC4to6HiV6YjLx.cph0gtNBtJ8P
```

L'ID di caricamento richiesto da questo comando viene emesso da `create-multipart-upload` e può essere recuperato anche con `list-multipart-uploads`.

- Per i dettagli sull'API, consulta [AbortMultipartUpload AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import software.amazon.awssdk.core.exception.SdkException;  
import software.amazon.awssdk.core.sync.RequestBody;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;  
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadResponse;  
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;  
import software.amazon.awssdk.services.s3.model.CompletedPart;  
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;  
import software.amazon.awssdk.services.s3.model.LifecycleRule;  
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;  
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;  
import software.amazon.awssdk.services.s3.model.MultipartUpload;  
import  
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationResponse;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import software.amazon.awssdk.services.s3.model.UploadPartRequest;  
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
```



```
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.utils.builder.SdkBuilder;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.time.Duration;
import java.time.Instant;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import java.util.Objects;
import java.util.UUID;

import static software.amazon.awssdk.transfer.s3.SizeConstant.KB;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class AbortMultipartUploadExamples {
    static final S3Client s3Client = S3Client.create();
    static final String classPathFilePath = "/multipartUploadFiles/s3-userguide.pdf";
    static final String filePath = getFullFilePath(classPathFilePath);
    private static final Logger logger =
        LoggerFactory.getLogger(AbortMultipartUploadExamples.class);
    private static final String accountId = getAccountId();

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.

        String key = UUID.randomUUID().toString();

        createBucket(bucketName);
    }
}
```

```

        try {
            initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread");
            abortIncompleteMultipartUploadsFromList(bucketName);
            abortMultipartUploadUsingUploadId(bucketName, key);
            abortMultipartUploadsUsingLifecycleConfig(bucketName);
        } catch (S3Exception e) {
            logger.error(e.getMessage());
        } finally {
            deleteResources(bucketName, key);
        }
    }

    /**
     * Aborts all incomplete multipart uploads from the specified S3 bucket.
     *
     * @param bucketName the name of the S3 bucket
     */
    public static void abortIncompleteMultipartUploadsFromList(String bucketName)
    {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
        ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        ListMultipartUploadsResponse response =
        s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(upload.key())
                .expectedBucketOwner(accountId)
                .uploadId(upload.uploadId())
                .build();

            AbortMultipartUploadResponse abortMultipartUploadResponse =
            s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                logger.info("Upload ID [{}] to bucket [{}] successfully
                aborted.", upload.uploadId(), bucketName);
            }
        }
    }

```

```

    }

    /**
     * Aborts incomplete multipart uploads older than the specified point in
     time.
     *
     * @param bucketName the name of the S3 bucket
     * @param pointInTime the cutoff time; uploads initiated before this are
     aborted
     */
    static void abortIncompleteMultipartUploadsOlderThan(String bucketName,
Instant pointInTime) {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            logger.info("Found multipartUpload with upload ID [{}], initiated
[{}]", upload.uploadId(), upload.initiated());
            if (upload.initiated().isBefore(pointInTime)) {
                abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(upload.key())
                    .expectedBucketOwner(accountId)
                    .uploadId(upload.uploadId())
                    .build();

                AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
                if
(abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                    logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
                }
            }
        }
    }
}

```

```

/**
 * Aborts a multipart upload using the upload ID.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 */
static void abortMultipartUploadUsingUploadId(String bucketName, String key)
{
    String uploadId = startUploadReturningUploadId(bucketName, key);
    AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b
        .uploadId(uploadId)
        .bucket(bucketName)
        .key(key));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
    }
}

/**
 * Configures a lifecycle rule to abort incomplete multipart uploads after 7
days.
 *
 * @param bucketName the name of the S3 bucket
 */
static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
    Collection<LifecycleRule> lifeCycleRules =
List.of(LifecycleRule.builder()
        .abortIncompleteMultipartUpload(b -> b.
            daysAfterInitiation(7))
        .status("Enabled")
        .filter(SdkBuilder::build) // Filter element is required.
        .build());

    PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
        .bucket(bucketName)
        .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

    if (response.sdkHttpResponse().isSuccessful()) {

```

```

        logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
    } else {
        logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
    }
}

/*****
Multipart upload methods
*****/

static void initiateAndInterruptMultiPartUpload(String bucketName, String
key, String threadName) {
    Runnable upload = () -> {
        try {
            doMultipartUpload(bucketName, key);
        } catch (SdkException e) {
            logger.error(e.getMessage());
        }
    };
    Thread uploadThread = new Thread(upload, threadName);
    uploadThread.start();
    try {
        Thread.sleep(Duration.ofSeconds(1).toMillis());
    } catch (InterruptedException e) {
        logger.error(e.getMessage());
    }
    uploadThread.interrupt();
}

static Instant initiateAndInterruptTwoUploads(String bucketName, String key)
{
    initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread1");
    try {
        Thread.sleep(Duration.ofSeconds(5).toMillis());
    } catch (InterruptedException e) {
        logger.error(e.getMessage());
    }
    Instant secondUploadInstant = Instant.now();
    initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread2");
    return secondUploadInstant;
}

```

```
static void doMultipartUpload(String bucketName, String key) {
    String uploadId = step1CreateMultipartUpload(bucketName, key);
    List<CompletedPart> completedParts = step2UploadParts(bucketName, key,
uploadId);
    step3CompleteMultipartUpload(bucketName, key, uploadId, completedParts);
}

static String step1CreateMultipartUpload(String bucketName, String key) {
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key));
    return createMultipartUploadResponse.uploadId();
}

static List<CompletedPart> step2UploadParts(String bucketName, String key,
String uploadId) {
    int partNumber = 1;
    List<CompletedPart> completedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(Long.valueOf(1024 * KB).intValue());

    try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
        long fileSize = file.length();
        long position = 0;
        while (position < fileSize) {
            file.seek(position);
            long read = file.getChannel().read(bb);

            bb.flip();
            UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .partNumber(partNumber)
                .build();

            UploadPartResponse partResponse = s3Client.uploadPart(
                uploadPartRequest,
                RequestBody.fromByteBuffer(bb));

            CompletedPart part = CompletedPart.builder()
                .partNumber(partNumber)
                .eTag(partResponse.eTag())
                .build();
        }
    }
```

```

        completedParts.add(part);
        logger.info("Part {} upload", partNumber);

        bb.clear();
        position += read;
        partNumber++;
    }
} catch (IOException | S3Exception e) {
    logger.error(e.getMessage());
    return null;
}
return completedParts;
}

static void step3CompleteMultipartUpload(String bucketName, String key,
String uploadId, List<CompletedPart> completedParts) {
    s3Client.completeMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}

static String startUploadReturningUploadId(String bucketName, String key) {
    String uploadId = step1CreateMultipartUpload(bucketName, key);
    doMultipartUploadWithUploadId(bucketName, key, uploadId);
    return uploadId;
}

static void doMultipartUploadWithUploadId(String bucketName, String key,
String uploadId) {
    new Thread(() -> {
        try {
            List<CompletedPart> completedParts = step2UploadParts(bucketName,
key, uploadId);
            step3CompleteMultipartUpload(bucketName, key, uploadId,
completedParts);
        } catch (SdkException e) {
            logger.error(e.getMessage());
        }
    }, "upload thread").start();
    try {
        Thread.sleep(Duration.ofSeconds(2L).toMillis());
    }
}

```

```

        } catch (InterruptedException e) {
            logger.error(e.getMessage());
            System.exit(1);
        }
    }

    /**
     * Resource handling methods
     */

    static void createBucket(String bucketName) {
        logger.info("Creating bucket: [{}]", bucketName);
        s3Client.createBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketExists(b -> b.bucket(bucketName));
        }
        logger.info("Bucket created.");
    }

    static void deleteResources(String bucketName, String key) {
        logger.info("Deleting resources ...");
        s3Client.deleteObject(b -> b.bucket(bucketName).key(key));
        s3Client.deleteBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketNotExists(b -> b.bucket(bucketName));
        }
        logger.info("Resources deleted.");
    }

    private static String getAccountId() {
        try (StsClient stsClient = StsClient.create()) {
            return stsClient.getCallerIdentity().account();
        }
    }

    static String getFullFilePath(String filePath) {
        URL uploadDirectoryURL =
PerformMultiPartUpload.class.getResource(filePath);
        String fullFilePath;
        try {
            fullFilePath =
Objects.requireNonNull(uploadDirectoryURL).toURI().getPath();
        } catch (URISyntaxException e) {
            throw new RuntimeException(e);
        }
    }

```



```
    }  
    return fullFilePath;  
  }  
}
```

- Per i dettagli sull'API, consulta la [AbortMultipartUpload](#) sezione AWS SDK for Java 2.x API Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando interrompe i caricamenti in più parti creati prima di 5 giorni fa.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

Esempio 2: questo comando interrompe i caricamenti in più parti creati prima del 2 gennaio 2014.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"Thursday, January 02, 2014"
```

Esempio 3: questo comando interrompe i caricamenti in più parti creati prima del 2 gennaio 2014 alle 10:45:37.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"2014/01/02 10:45:37"
```

- Per i dettagli sull'API, vedere [AbortMultipartUpload](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando interrompe i caricamenti in più parti creati prima di 5 giorni fa.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

Esempio 2: questo comando interrompe i caricamenti in più parti creati prima del 2 gennaio 2014.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"Thursday, January 02, 2014"
```

Esempio 3: questo comando interrompe i caricamenti in più parti creati prima del 2 gennaio 2014 alle 10:45:37.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"2014/01/02 10:45:37"
```

- Per i dettagli sull'API, vedere [AbortMultipartUpload](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **CompleteMultipartUpload** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare CompleteMultipartUpload.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- [Eseguire una copia in più parti](#)
- [Utilizzo dei checksum](#)
- [Utilizzare la funzionalità per l'integrità degli oggetti di Amazon S3](#)

C++

SDK per C++

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
//! Complete a multipart upload to an S3 bucket.
```

```

/ * !
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

    const Aws::String &key,

    const Aws::String &uploadID,

    const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

    const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
        client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}

```

- Per i dettagli sull'API, consulta la [CompleteMultipartUpload](#) sezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Il comando seguente completa un caricamento in più parti per la chiave `multipart/01` nel bucket `amzn-s3-demo-bucket`.

```
aws s3api complete-multipart-upload --multipart-upload file://  
mpustruct --bucket amzn-s3-demo-bucket --key 'multipart/01' --upload-  
id dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6YjlX.cph0gtNBtJ8P
```

L'ID di caricamento richiesto da questo comando viene emesso da `create-multipart-upload` e può essere recuperato anche con `list-multipart-uploads`.

L'opzione di caricamento in più parti utilizzato nel precedente comando utilizza una struttura JSON che descrive le parti del caricamento in più parti che devono essere riassemblate nel file completo. In questo esempio, il prefisso `file://` è usato per caricare la struttura JSON da un file nella cartella locale chiamato `mpustruct`.

`mpustruct`:

```
{  
  "Parts": [  
    {  
      "ETag": "e868e0f4719e394144ef36531ee6824c",  
      "PartNumber": 1  
    },  
    {  
      "ETag": "6bb2b12753d66fe86da4998aa33fffb0",  
      "PartNumber": 2  
    },  
    {  
      "ETag": "d0a0112e841abec9c9ec83406f0159c8",  
      "PartNumber": 3  
    }  
  ]  
}
```

Il valore ETag per ogni parte del caricamento viene restituito ogni volta che si carica una parte utilizzando il comando `upload-part`. Tale valore può essere recuperato anche chiamando `list-parts` o venire calcolato utilizzando il checksum MD5 di ciascuna parte.

Output:

```
{
  "ETag": "\"3944a9f7a4faab7f78788ff6210f63f0-3\"",
  "Bucket": "amzn-s3-demo-bucket",
  "Location": "https://amzn-s3-demo-bucket.s3.amazonaws.com/multipart%2F01",
  "Key": "multipart/01"
}
```

- Per i dettagli sull'API, consulta [CompleteMultipartUpload AWS CLI](#) Command Reference.

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;
```

```
// Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
// upload the file.
let multipart_upload_res: CreateMultipartUploadOutput = client
```

```

        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;

let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
    "Missing upload_id after CreateMultipartUpload",
))?;

```

```

let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
    let upload_part_res = client
        .upload_part()
        .key(&key)
        .bucket(&bucket_name)
        .upload_id(upload_id)
        .body(stream)
        .part_number(part_number)
        .send()
        .await?;

    upload_parts.push(
        CompletedPart::builder()
            .e_tag(upload_part_res.e_tag.unwrap_or_default())
            .part_number(part_number)

```

```
        .build(),  
    );  
}
```

- Per i dettagli sulle API, consulta la [CompleteMultipartUpload](#) guida di riferimento all'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **CopyObject** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare CopyObject.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- [Informazioni di base](#)
- [Nozioni di base sulla crittografia](#)
- [Guida introduttiva allo storage di oggetti](#)
- [Eseguire richieste condizionali](#)

.NET

SDK per .NET (v4)



Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>  
/// Copies an object in an Amazon S3 bucket to a folder within the  
/// same bucket.
```

```
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket where the
/// object to copy is located.</param>
/// <param name="objectName">The object to be copied.</param>
/// <param name="folderName">The folder to which the object will
/// be copied.</param>
/// <returns>A boolean value that indicates the success or failure of
/// the copy operation.</returns>
public async Task<bool> CopyObjectInBucketAsync(
    string bucketName,
    string objectName,
    string folderName)
{
    try
    {
        var request = new CopyObjectRequest
        {
            SourceBucket = bucketName,
            SourceKey = objectName,
            DestinationBucket = bucketName,
            DestinationKey = $"{folderName}\\{objectName}",
        };
        var response = await _amazonS3.CopyObjectAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error copying object: '{ex.Message}'");
        return false;
    }
}
```

- Per i dettagli sull'API, consulta la [CopyObject](#) sezione AWS SDK per .NET API Reference.

SDK per .NET

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Copia un oggetto utilizzando una richiesta condizionale.

```
/// <summary>
/// Copies an object from one Amazon S3 bucket to another with a conditional
request.
/// </summary>
/// <param name="sourceKey">The key of the source object to copy.</param>
/// <param name="destKey">The key of the destination object.</param>
/// <param name="sourceBucket">The source bucket of the object.</param>
/// <param name="destBucket">The destination bucket of the object.</param>
/// <param name="conditionType">The type of condition to apply, e.g.
'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
'CopySourceIfUnmodifiedSince'.</param>
/// <param name="conditionDateValue">The value to use for the condition for
dates.</param>
/// <param name="etagConditionalValue">The value to use for the condition for
etags.</param>
/// <returns>True if the conditional copy is successful, False otherwise.</
returns>
public async Task<bool> CopyObjectConditional(string sourceKey, string
destKey, string sourceBucket, string destBucket,
    S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
{
    try
    {
        var copyObjectRequest = new CopyObjectRequest
        {
            DestinationBucket = destBucket,
            DestinationKey = destKey,
            SourceBucket = sourceBucket,
            SourceKey = sourceKey
        };

        switch (conditionType)
        {
            case S3ConditionType.IfMatch:
                copyObjectRequest.ETagToMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfNoneMatch:
                copyObjectRequest.ETagToNotMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfModifiedSince:
```

```

        copyObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
        break;
        case S3ConditionType.IfUnmodifiedSince:
            copyObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
            break;
        default:
            throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
    }

    await _amazonS3.CopyObjectAsync(copyObjectRequest);
    _logger.LogInformation($"Conditional copy successful for key
{destKey} in bucket {destBucket}.");
    return true;
}
catch (AmazonS3Exception e)
{
    if (e.ErrorCode == "PreconditionFailed")
    {
        _logger.LogError("Conditional copy failed: Precondition failed");
    }
    else if (e.ErrorCode == "304")
    {
        _logger.LogError("Conditional copy failed: Object not modified");
    }
    else
    {
        _logger.LogError($"Unexpected error: {e.ErrorCode}");
        throw;
    }
    return false;
}
}

```

- Per i dettagli sull'API, consulta la [CopyObject](#) sezione AWS SDK per .NET API Reference.

Bash

AWS CLI con lo script Bash

 Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file from and to.
#     $2 - The key of the source file to copy.
#     $3 - The key of the destination file.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function copy_item_in_bucket() {
    local bucket_name=$1
    local source_key=$2
    local destination_key=$3
    local response

    response=$(aws s3api copy-object \
        --bucket "$bucket_name" \
        --copy-source "$bucket_name/$source_key" \
```

```
--key "$destination_key")

# shellcheck disable=SC2181
if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api copy-object operation failed.\n$response"
    return 1
fi
}
```

- Per i dettagli sull'API, consulta [CopyObject AWS CLI Command Reference](#).

C++

SDK per C++

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
bool AwsDoc::S3::copyObject(const Aws::String &objectKey, const Aws::String
&fromBucket, const Aws::String &toBucket,
                        const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CopyObjectRequest request;

    request.WithCopySource(fromBucket + "/" + objectKey)
        .WithKey(objectKey)
        .WithBucket(toBucket);

    Aws::S3::Model::CopyObjectOutcome outcome = client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: copyObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
        std::endl;
    } else {
        std::cout << "Successfully copied " << objectKey << " from " <<
            fromBucket <<
```

```
        " to " << toBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, consulta la [CopyObject](#) sezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Il comando seguente copia un oggetto da bucket-1 a bucket-2.

```
aws s3api copy-object --copy-source bucket-1/test.txt --key test.txt --
bucket bucket-2
```

Output:

```
{
  "CopyObjectResult": {
    "LastModified": "2015-11-10T01:07:25.000Z",
    "ETag": "\"589c8b79c230a6ecd5a7e1d040a9a030\""
  },
  "VersionId": "YdnYvTCVDqRRFA.NFJjy36p0hxifM1kA"
}
```

- Per i dettagli sull'API, consulta [CopyObject AWS CLI](#) Command Reference.

Go

SDK per Go V2

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "io"
    "log"
    "os"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
    destinationBucket string, objectKey string) error {
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
        Bucket:      aws.String(destinationBucket),
        CopySource:  aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
        Key:         aws.String(objectKey),
    })
    if err != nil {
        var notActive *types.ObjectNotInActiveTierError
        if errors.As(err, &notActive) {
            log.Printf("Couldn't copy object %s from %s because the object isn't in the
            active tier.\n",
```

```

    objectKey, sourceBucket)
    err = notActive
}
} else {
    err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
        ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
aws.String(objectKey)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
    }
}
return err
}

```

- Per i dettagli sull'API, consulta la [CopyObject](#) sezione AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Copia un oggetto usando un'interfaccia [S3Client](#).

```

/**
 * Asynchronously copies an object from one S3 bucket to another.
 *
 * @param fromBucket the name of the source S3 bucket
 * @param objectKey the key (name) of the object to be copied
 * @param toBucket the name of the destination S3 bucket
 * @return a {@link CompletableFuture} that completes with the copy result as
a {@link String}
 * @throws RuntimeException if the URL could not be encoded or an S3
exception occurred during the copy
 */

```

```

    public CompletableFuture<String> copyBucketObjectAsync(String fromBucket,
String objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(fromBucket)
            .sourceKey(objectKey)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();

        CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
        response.whenComplete((copyRes, ex) -> {
            if (copyRes != null) {
                logger.info("The " + objectKey + " was copied to " + toBucket);
            } else {
                throw new RuntimeException("An S3 exception occurred during
copy", ex);
            }
        });

        return response.thenApply(CopyObjectResponse::copyObjectResult)
            .thenApply(Object::toString);
    }

```

Usa un [S3TransferManager](#) per [copiare un oggetto](#) da un bucket all'altro. Visualizza il [file completo](#) ed esegui il [test](#).

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

    public String copyObject(S3TransferManager transferManager, String
bucketName,
        String key, String destinationBucket, String destinationKey) {
        CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()

```



```

        .sourceBucket(bucketName)
        .sourceKey(key)
        .destinationBucket(destinationBucket)
        .destinationKey(destinationKey)
        .build();

CopyRequest copyRequest = CopyRequest.builder()
    .copyObjectRequest(copyObjectRequest)
    .build();

Copy copy = transferManager.copy(copyRequest);

CompletedCopy completedCopy = copy.completionFuture().join();
return completedCopy.response().copyObjectResult().eTag();
}

```

- Per i dettagli sull'API, consulta la sezione [CopyObject AWS SDK for Java 2.x API Reference](#).

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Copia l'oggetto.

```

import {
    S3Client,
    CopyObjectCommand,
    ObjectNotInActiveTierError,
    waitUntilObjectExists,
} from "@aws-sdk/client-s3";

/**
 * Copy an S3 object from one bucket to another.
 *
 * @param {{

```

```
*   sourceBucket: string,
*   sourceKey: string,
*   destinationBucket: string,
*   destinationKey: string }} config
*/
export const main = async ({
  sourceBucket,
  sourceKey,
  destinationBucket,
  destinationKey,
}) => {
  const client = new S3Client({});

  try {
    await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucket}/${sourceKey}`,
        Bucket: destinationBucket,
        Key: destinationKey,
      }),
    );
    await waitUntilObjectExists(
      { client },
      { Bucket: destinationBucket, Key: destinationKey },
    );
    console.log(
      `Successfully copied ${sourceBucket}/${sourceKey} to ${destinationBucket}/${destinationKey}`,
    );
  } catch (caught) {
    if (caught instanceof ObjectNotInActiveTierError) {
      console.error(
        `Could not copy ${sourceKey} from ${sourceBucket}. Object is not in the active tier.`,
      );
    } else {
      throw caught;
    }
  }
};
```

Copia l'oggetto a condizione che il relativo ETag non corrisponda a quello fornito.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
  destinationBucketName: string, eTag: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
  eTag,
}) => {
  const client = new S3Client({});
  const name = data.name;
  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucketName}/${sourceKeyName}`,
        Bucket: destinationBucketName,
        Key: `${name}${sourceKeyName}`,
        CopySourceIfMatch: eTag,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
        "${sourceBucketName}". No such key exists.`
      );
    }
  }
};
```

```
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Unable to copy object "${sourceKeyName}" to bucket
        "${sourceBucketName}": ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
```

```
if (!errors) {
  main(results.values);
} else {
  console.error(errors.join("\n"));
}
}
```

Copia l'oggetto a condizione che il relativo ETag non corrisponda a quello fornito.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
  destinationBucketName: string, eTag: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
  eTag,
}) => {
  const client = new S3Client({});
  const name = data.name;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucketName}/${sourceKeyName}`,
        Bucket: destinationBucketName,
```

```

        Key: `${name}${sourceKeyName}`,
        CopySourceIfNoneMatch: eTag,
    )),
    );
    console.log("Successfully copied object to bucket.");
} catch (caught) {
    if (caught instanceof NoSuchKey) {
        console.error(
            `Error from S3 while copying object "${sourceKeyName}" from
"${sourceBucketName}". No such key exists.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Unable to copy object "${sourceKeyName}" to bucket
"${sourceBucketName}": ${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
    isMain,
    validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
    const options = {
        sourceBucketName: {
            type: "string",
            required: true,
        },
        sourceKeyName: {
            type: "string",
            required: true,
        },
        destinationBucketName: {
            type: "string",
            required: true,
        },
        eTag: {

```

```

        type: "string",
        required: true,
    },
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
        main(results.values);
    } else {
        console.error(errors.join("\n"));
    }
}
}

```

Copia l'oggetto a condizione che sia stato creato o modificato in un determinato periodo di tempo.

```

import {
    CopyObjectCommand,
    NoSuchKey,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
    type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
    destinationBucketName: string }}
 */
export const main = async ({
    sourceBucketName,

```

```
    sourceKeyName,
    destinationBucketName,
  }) => {
    const date = new Date();
    date.setDate(date.getDate() - 1);

    const name = data.name;
    const client = new S3Client({});
    const copySource = `${sourceBucketName}/${sourceKeyName}`;
    const copiedKey = name + sourceKeyName;

    try {
      const response = await client.send(
        new CopyObjectCommand({
          CopySource: copySource,
          Bucket: destinationBucketName,
          Key: copiedKey,
          CopySourceIfModifiedSince: date,
        }),
      );
      console.log("Successfully copied object to bucket.");
    } catch (caught) {
      if (caught instanceof NoSuchKey) {
        console.error(
          `Error from S3 while copying object "${sourceKeyName}" from
"${sourceBucketName}". No such key exists.`,
        );
      } else if (caught instanceof S3ServiceException) {
        console.error(
          `Error from S3 while copying object from ${sourceBucketName}.
${caught.name}: ${caught.message}`,
        );
      } else {
        throw caught;
      }
    }
  };

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";
```



```
const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Copia l'oggetto a condizione che non sia stato creato o modificato in un determinato periodo di tempo.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";
```

```
// Optionally edit the default key name of the copied object in
'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
  destinationBucketName: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
}) => {
  const date = new Date();
  date.setDate(date.getDate() - 1);
  const client = new S3Client({});
  const name = data.name;
  const copiedKey = name + sourceKeyName;
  const copySource = `${sourceBucketName}/${sourceKeyName}`;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: copySource,
        Bucket: destinationBucketName,
        Key: copiedKey,
        CopySourceIfUnmodifiedSince: date,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
        "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while copying object from ${sourceBucketName}.
        ${caught.name}: ${caught.message}`,
      );
    }
  }
};
```

```
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
  };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Per i dettagli sull'API, consulta la [CopyObject](#) sezione AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedOperationException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- Per i dettagli sull'API, [CopyObject](#) consulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Copia semplice di un oggetto.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $folder = "copied-folder";
    $this->s3client->copyObject([
        'Bucket' => $this->bucketName,
        'CopySource' => "$this->bucketName/$fileName",
        'Key' => "$folder/$fileName-copy",
    ]);
    echo "Copied $fileName to $folder/$fileName-copy.\n";
} catch (Exception $exception) {
    echo "Failed to copy $fileName with error: " . $exception-
    >getMessage();
    exit("Please fix error with object copying before continuing.");
}
```

- Per i dettagli sull'API, consulta la [CopyObject](#) sezione AWS SDK per PHP API Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando copia l'oggetto "sample.txt" dal bucket "test-files" nello stesso bucket ma con una nuova chiave "sample-copy.txt".

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey
sample-copy.txt
```

Esempio 2: questo comando copia l'oggetto "sample.txt" dal bucket "test-files" nel bucket "backup-files" con una chiave "sample-copy.txt".

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -  
DestinationKey sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

Esempio 3: questo comando scarica l'oggetto "sample.txt" dal bucket "test-files" in un file locale denominato "local-sample.txt".

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-  
sample.txt
```

Esempio 4: scarica un unico oggetto nel file specificato. Il file scaricato si trova in c:\downloads\data/archive.zip

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder  
c:\downloads
```

Esempio 5: scarica tutti gli oggetti che corrispondono al prefisso di chiave specificato nella cartella locale. La corrispondente gerarchia di chiavi verrà conservata come sottocartelle nella posizione generale di scaricamento.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:  
\downloads
```

- Per i dettagli sull'API, vedere [CopyObject](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando copia l'oggetto "sample.txt" dal bucket "test-files" nello stesso bucket ma con una nuova chiave "sample-copy.txt".

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey  
sample-copy.txt
```

Esempio 2: questo comando copia l'oggetto "sample.txt" dal bucket "test-files" nel bucket "backup-files" con una chiave "sample-copy.txt".

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -  
DestinationKey sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

Esempio 3: questo comando scarica l'oggetto "sample.txt" dal bucket "test-files" in un file locale denominato "local-sample.txt".

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-  
sample.txt
```

Esempio 4: scarica un unico oggetto nel file specificato. Il file scaricato si trova in c:\downloads\data\archive.zip

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder  
c:\downloads
```

Esempio 5: scarica tutti gli oggetti che corrispondono al prefisso di chiave specificato nella cartella locale. La corrispondente gerarchia di chiavi verrà conservata come sottocartelle nella posizione generale di scaricamento.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:  
\downloads
```

Esempio 6: scarica un singolo oggetto in un file locale utilizzando il download parallelo multipart per migliorare le prestazioni su file di grandi dimensioni.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-  
sample.txt -UseMultipartDownload
```

Esempio 7: scarica un oggetto di grandi dimensioni utilizzando la modalità RANGE con una dimensione di parte personalizzata di 64 MB. La modalità RANGE utilizza richieste HTTP con intervallo di byte e funziona per tutti gli oggetti indipendentemente dal modo in cui sono stati caricati.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile  
local-large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -  
PartSize 64MB
```

Esempio 8: scarica tutti gli oggetti che corrispondono al prefisso del key prefix in una cartella locale utilizzando il download in più parti con download simultanei di file.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:\local-data -UseMultipartDownload -DownloadFilesConcurrently
```

Esempio 9: scarica un singolo oggetto utilizzando il download in più parti con 20 connessioni simultanee per la massima velocità di trasmissione su reti a larghezza di banda elevata.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile local-large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

Esempio 10: scarica tutti gli oggetti che corrispondono al prefisso del key prefix in una cartella locale utilizzando il download in più parti, continuando a superare gli errori dei singoli file invece di interrompere.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:\local-data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- Per i dettagli sull'API, vedere [CopyObject](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Python

SDK per Python (Boto3)

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
```



```

        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                                that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
key.
                                This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
self.object.key}
            )
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
        except ClientError:
            logger.exception(
                "Couldn't copy object from %s/%s to %s/%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
            raise

```

Copia un oggetto utilizzando una richiesta condizionale.

```

class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def copy_object_conditional(
        self,
        source_key: str,
        dest_key: str,
        source_bucket: str,
        dest_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Copies an object from one Amazon S3 bucket to another with a conditional
        request.

        :param source_key: The key of the source object to copy.
        :param dest_key: The key of the destination object.
        :param source_bucket: The source bucket of the object.
        :param dest_bucket: The destination bucket of the object.
        :param condition_type: The type of condition to apply, e.g.
            'CopySourceIfMatch', 'CopySourceIfNoneMatch',
            'CopySourceIfModifiedSince', 'CopySourceIfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            self.s3.copy_object(
                Bucket=dest_bucket,
                Key=dest_key,
                CopySource={"Bucket": source_bucket, "Key": source_key},
                **{condition_type: condition_value},

```

```

    )
    print(
        f"\tConditional copy successful for key {dest_key} in bucket
{dest_bucket}."
    )
except ClientError as e:
    error_code = e.response["Error"]["Code"]
    if error_code == "PreconditionFailed":
        print("\tConditional copy failed: Precondition failed")
    elif error_code == "304": # Not modified error code.
        print("\tConditional copy failed: Object not modified")
    else:
        logger.error(f"Unexpected error: {error_code}")
        raise

```

- Per i dettagli sull'API, consulta [CopyObject AWS SDK for Python \(Boto3\) API Reference](#).

Ruby

SDK per Ruby

Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Copia un oggetto.

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
  used as the source object for
  #
  #
  def initialize(source_object)
    @source_object = source_object
  end
end

```

```

end

# Copy the source object to the specified target bucket and rename it with the
target key.
#
# @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
object is copied.
# @param target_object_key [String] The key to give the copy of the object.
# @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
nil.
def copy_object(target_bucket, target_object_key)
  @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
  target_bucket.object(target_object_key)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
why: #{e.message}"
end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
#{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__

```

Copia un oggetto e aggiungi la crittografia lato server all'oggetto di destinazione.

```
require 'aws-sdk-s3'
```

```

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
  # used as the source object for
  #                                     copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the
  # target key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
  # object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
  # nil.
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
why: #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key,
target_encryption)
  return unless target_object
end

```

```
puts "Copied #{source_key} from #{source_bucket_name} to
#{target_object.bucket_name}:#{target_object.key} and "\
    "encrypted the target with #{target_object.server_side_encryption}
encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, consulta la [CopyObject](#) sezione AWS SDK per Ruby API Reference.

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// Copy an object from one bucket to another.
pub async fn copy_object(
    client: &aws_sdk_s3::Client,
    source_bucket: &str,
    destination_bucket: &str,
    source_object: &str,
    destination_object: &str,
) -> Result<(), S3ExampleError> {
    let source_key = format!("{source_bucket}/{source_object}");
    let response = client
        .copy_object()
        .copy_source(&source_key)
        .bucket(destination_bucket)
        .key(destination_object)
        .send()
        .await?;

    println!(
        "Copied from {source_key} to {destination_bucket}/{destination_object}
with etag {}",
        response
```

```

        .copy_object_result
        .unwrap_or_else(||
aws_sdk_s3::types::CopyObjectResult::builder().build())
        .e_tag()
        .unwrap_or("missing")
    );
    Ok(())
}

```

- Per i dettagli sulle API, consulta la [CopyObject](#) guida di riferimento all'API AWS SDK for Rust.

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

TRY.
    lo_s3->copyobject(
        iv_bucket = iv_dest_bucket
        iv_key = iv_dest_object
        iv_copysource = |{ iv_src_bucket }/{ iv_src_object }| ).
    MESSAGE 'Object copied to another bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

```

- Per i dettagli sulle API, [CopyObject](#) consulta AWS SDK for SAP ABAP API reference.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import AWSS3

public func copyFile(from sourceBucket: String, name: String, to destBucket:
String) async throws {
    let srcUrl = ("\"(sourceBucket)/
\"(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

    let input = CopyObjectInput(
        bucket: destBucket,
        copySource: srcUrl,
        key: name
    )
    do {
        _ = try await client.copyObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Copying an object."))
        throw error
    }
}
```

- Per i dettagli sull'API, consulta la [CopyObject](#)guida di riferimento all'API AWS SDK for Swift.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **CreateBucket** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare CreateBucket.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- [Informazioni di base](#)
- [Guida introduttiva alla gestione della configurazione](#)
- [Guida introduttiva agli store di funzionalità di apprendimento automatico](#)
- [Guida introduttiva allo storage di oggetti](#)
- [Gestire messaggi di grandi dimensioni con S3](#)
- [Utilizzo degli oggetti con versione](#)

.NET

SDK per .NET (v4)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>
/// Shows how to create a new Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <returns>A boolean value representing the success or failure of
/// the bucket creation process.</returns>
public async Task<bool> CreateBucketAsync(string bucketName)
{
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true,
        };

        var response = await _amazonS3.PutBucketAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
}
```

```
    }  
    catch (AmazonS3Exception ex)  
    {  
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");  
        return false;  
    }  
}
```

- Per i dettagli sull'API, consulta la [CreateBucket](#) sezione AWS SDK per .NET API Reference. SDK per .NET

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea un bucket con Object Lock abilitato.

```
/// <summary>  
/// Create a new Amazon S3 bucket with object lock actions.  
/// </summary>  
/// <param name="bucketName">The name of the bucket to create.</param>  
/// <param name="enableObjectLock">True to enable object lock on the  
bucket.</param>  
/// <returns>True if successful.</returns>  
public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool  
enableObjectLock)  
{  
    Console.WriteLine($"\\tCreating bucket {bucketName} with object lock  
{enableObjectLock}.");  
    try  
    {  
        var request = new PutBucketRequest  
        {  
            BucketName = bucketName,  
            UseClientRegion = true,  
            ObjectLockEnabledForBucket = enableObjectLock,  
        };  
    }  
}
```

```

        var response = await _amazonS3.PutBucketAsync(request);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
        return false;
    }
}

```

- Per i dettagli sull'API, consulta la [CreateBucket](#) sezione AWS SDK per .NET API Reference.

Bash

AWS CLI con lo script Bash

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {

```

```

    printf "%s\n" "$*" 1>&2
}

#####
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#     -b bucket_name  -- The name of the bucket to create.
#     -r region_code  -- The code for an AWS Region in which to
#                        create the bucket.
#
# Returns:
#     The URL of the bucket that was created.
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function create_bucket() {
    local bucket_name region_code response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function create_bucket"
        echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
        echo "  -b bucket_name    The name of the bucket. It must be globally
unique."
        echo "  [-r region_code]    The code for an AWS Region in which the bucket is
created."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "b:r:h" option; do
        case "${option}" in
            b) bucket_name="${OPTARG}" ;;
            r) region_code="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
        esac
    done
}

```

```
\?)
    echo "Invalid parameter"
    usage
    return 1
    ;;
esac
done

if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
fi

local bucket_config_arg
# A location constraint for "us-east-1" returns an error.
if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
fi

iecho "Parameters:\n"
iecho "    Bucket name:  $bucket_name"
iecho "    Region code:  $region_code"
iecho ""

# If the bucket already exists, we don't want to try to create it.
if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
fi

# shellcheck disable=SC2086
response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
fi
}
```

- Per i dettagli sull'API, consulta [CreateBucket AWS CLI Command Reference](#).

C++

SDK per C++

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
bool AwsDoc::S3::createBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(
            Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
            std::endl;
    } else {
        std::cout << "Created bucket " << bucketName <<
            " in the specified AWS Region." << std::endl;
    }

    return outcome.IsSuccess();
}
```

```
}
```

- Per i dettagli sull'API, consulta la [CreateBucket](#) sezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Esempio 1: come creare un bucket

L'esempio create-bucket seguente crea un bucket denominato amzn-s3-demo-bucket.

```
aws s3api create-bucket \
  --bucket amzn-s3-demo-bucket \
  --region us-east-1
```

Output:

```
{
  "Location": "/amzn-s3-demo-bucket"
}
```

Per ulteriori informazioni, consulta [Creazione di un bucket](#) nella Guida per l'utente di Amazon S3.

Esempio 2: come creare un bucket con il proprietario impostato

L'esempio create-bucket seguente crea un bucket denominato amzn-s3-demo-bucket che utilizza l'impostazione Proprietario del bucket applicato per la proprietà degli oggetti S3.

```
aws s3api create-bucket \
  --bucket amzn-s3-demo-bucket \
  --region us-east-1 \
  --object-ownership BucketOwnerEnforced
```

Output:

```
{
  "Location": "/amzn-s3-demo-bucket"
}
```

Per ulteriori informazioni, consulta [Controllo della proprietà degli oggetti e disabilitazione degli ACL](#) nella Guida per l'utente di Amazon S3.

Esempio 3: come creare un bucket al di fuori della Regione "us-east

L'esempio create-bucket seguente crea un bucket denominato amzn-s3-demo-bucket nella Regione eu-west-1. Le Regioni esterne a us-east-1 richiedono la specifica del valore appropriato per LocationConstraint affinché il bucket venga creato nella Regione desiderata.

```
aws s3api create-bucket \  
  --bucket amzn-s3-demo-bucket \  
  --region eu-west-1 \  
  --create-bucket-configuration LocationConstraint=eu-west-1
```

Output:

```
{  
  "Location": "http://amzn-s3-demo-bucket.s3.amazonaws.com/"  
}
```

Per ulteriori informazioni, consulta [Creazione di un bucket](#) nella Guida per l'utente di Amazon S3.

- Per i dettagli sull'API, consulta [CreateBucket AWS CLI](#) Command Reference.

Go

SDK per Go V2

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea un bucket con la configurazione predefinita.

```
import (  
  "bytes"
```



```
"context"
"errors"
"fmt"
"io"
"log"
"os"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
    string) error {
    _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
        Bucket: aws.String(name),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    })
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", name)
            err = owned
        } else if errors.As(err, &exists) {
            log.Printf("Bucket %s already exists.\n", name)
```

```
    err = exists
  }
} else {
  err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
    ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
  if err != nil {
    log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
  }
}
return err
}
```

Crea un bucket con Object Lock abilitato e attende la verifica della sua esistenza.

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// CreateBucketWithLock creates a new S3 bucket with optional object locking
// enabled
// and waits for the bucket to exist before returning.
```

```
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
    region string, enableObjectLock bool) (string, error) {
    input := &s3.CreateBucketInput{
        Bucket: aws.String(bucket),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    }

    if enableObjectLock {
        input.ObjectLockEnabledForBucket = aws.Bool(true)
    }

    _, err := actor.S3Client.CreateBucket(ctx, input)
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", bucket)
            err = owned
        } else if errors.As(err, &exists) {
            log.Printf("Bucket %s already exists.\n", bucket)
            err = exists
        }
    } else {
        err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
        }
    }

    return bucket, err
}
```

- Per i dettagli sull'API, consulta la [CreateBucket](#) sezione AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare un bucket.

```
/**
 * Creates an S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to create
 * @return a {@link CompletableFuture} that completes when the bucket is
 *         created and ready
 * @throws RuntimeException if there is a failure while creating the bucket
 */
public CompletableFuture<Void> createBucketAsync(String bucketName) {
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .build();

    CompletableFuture<CreateBucketResponse> response =
        getAsyncClient().createBucket(bucketRequest);
    return response.thenCompose(resp -> {
        S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        CompletableFuture<WaiterResponse<HeadBucketResponse>>
        waiterResponseFuture =
            s3Waiter.waitUntilBucketExists(bucketRequestWait);
        return waiterResponseFuture.thenAccept(waiterResponse -> {
            waiterResponse.matched().response().ifPresent(headBucketResponse
-> {
                logger.info(bucketName + " is ready");
            });
        });
    });
}
```

```
    }).whenComplete((resp, ex) -> {  
        if (ex != null) {  
            throw new RuntimeException("Failed to create bucket", ex);  
        }  
    });  
}
```

Crea un bucket con Object Lock abilitato.

```
// Create a new Amazon S3 bucket with object lock options.  
public void createBucketWithLockOptions(boolean enableObjectLock, String  
bucketName) {  
    S3Waiter s3Waiter = getClient().waiter();  
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()  
        .bucket(bucketName)  
        .objectLockEnabledForBucket(enableObjectLock)  
        .build();  
  
    getClient().createBucket(bucketRequest);  
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()  
        .bucket(bucketName)  
        .build();  
  
    // Wait until the bucket is created and print out the response.  
    s3Waiter.waitUntilBucketExists(bucketRequestWait);  
    System.out.println(bucketName + " is ready");  
}
```

- Per i dettagli sull'API, consulta la [CreateBucket](#) sezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea il bucket.

```
import {
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  CreateBucketCommand,
  S3Client,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

/**
 * Create an Amazon S3 bucket.
 * @param {{ bucketName: string }} config
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { Location } = await client.send(
      new CreateBucketCommand({
        // The name of the bucket. Bucket names are unique and have several other
        // constraints.
        // See https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucketnamingrules.html
        Bucket: bucketName,
      }),
    );
    await waitUntilBucketExists({ client }, { Bucket: bucketName });
    console.log(`Bucket created with location ${Location}`);
  } catch (caught) {
    if (caught instanceof BucketAlreadyExists) {
      console.error(
        `The bucket "${bucketName}" already exists in another AWS account. Bucket names must be globally unique.`,
      );
    }
    // WARNING: If you try to create a bucket in the North Virginia region,
    // and you already own a bucket in that region with the same name, this
    // error will not be thrown. Instead, the call will return successfully
    // and the ACL on that bucket will be reset.
    else if (caught instanceof BucketAlreadyOwnedByYou) {
      console.error(
        `The bucket "${bucketName}" already exists in this AWS account.`,
      );
    }
  }
}
```

```
    } else {  
        throw caught;  
    }  
}  
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, consulta la [CreateBucket](#) sezione AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun createNewBucket(bucketName: String) {  
    val request =  
        CreateBucketRequest {  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.createBucket(request)  
        println("$bucketName is ready")  
    }  
}
```

- Per i dettagli sull'API, [CreateBucket](#) consulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creare un bucket.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $this->s3client->createBucket([
        'Bucket' => $this->bucketName,
        'CreateBucketConfiguration' => ['LocationConstraint' => $region],
    ]);
    echo "Created bucket named: $this->bucketName \n";
} catch (Exception $exception) {
    echo "Failed to create bucket $this->bucketName with error: " .
    $exception->getMessage();
    exit("Please fix error with bucket creation before continuing.");
}
```

- Per i dettagli sull'API, consulta la [CreateBucket](#) sezione AWS SDK per PHP API Reference.

Python

SDK per Python (Boto3)

Note

C'è altro su [GitHub](#). Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea un bucket con le impostazioni di default.


```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def create(self, region_override=None):
        """
        Create an Amazon S3 bucket in the default Region for the account or in
        the
        specified Region.

        :param region_override: The Region in which to create the bucket. If this
        is
                                not specified, the Region configured in your
        shared
                                credentials is used.
        """
        if region_override is not None:
            region = region_override
        else:
            region = self.bucket.meta.client.meta.region_name
        try:
            self.bucket.create(CreateBucketConfiguration={"LocationConstraint":
            region})

            self.bucket.wait_until_exists()
            logger.info("Created bucket '%s' in region=%s", self.bucket.name,
            region)
        except ClientError as error:
            logger.exception(
                "Couldn't create bucket named '%s' in region=%s.",
                self.bucket.name,
                region,
            )
            raise error

```

Crea un bucket con versione con una configurazione del ciclo di vita.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
    lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                    configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
```

```
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket
```

- Per i dettagli sull'API, consulta [CreateBucket AWS SDK for Python \(Boto3\) API Reference](#).

Ruby

SDK per Ruby

Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
  # This is a client-side object until
  # create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      'None. You must create a bucket before you can get its location!'
    else

```

```

        @bucket.client.get_bucket_location(bucket:
@bucket.name).location_constraint
      end
      rescue Aws::Errors::ServiceError => e
        "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
      end
    end
  end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("amzn-s3-demo-bucket-
#{@Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{@wrapper.bucket.name}."
  puts "Your bucket's region is: #{@wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__

```

- Per i dettagli sull'API, consulta la [CreateBucket](#) sezione AWS SDK per Ruby API Reference.

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

pub async fn create_bucket(
  client: &aws_sdk_s3::Client,
  bucket_name: &str,
  region: &aws_config::Region,
) -> Result<Option<aws_sdk_s3::operation::create_bucket::CreateBucketOutput>,
S3ExampleError> {
  let constraint =
aws_sdk_s3::types::BucketLocationConstraint::from(region.to_string().as_str());

```

```
let cfg = aws_sdk_s3::types::CreateBucketConfiguration::builder()
    .location_constraint(constraint)
    .build();
let create = client
    .create_bucket()
    .create_bucket_configuration(cfg)
    .bucket(bucket_name)
    .send()
    .await;

// BucketAlreadyExists and BucketAlreadyOwnedByYou are not problems for this
task.
create.map(Some).or_else(|err| {
    if err
        .as_service_error()
        .map(|se| se.is_bucket_already_exists() ||
se.is_bucket_already_owned_by_you())
        == Some(true)
    {
        Ok(None)
    } else {
        Err(S3ExampleError::from(err))
    }
}))
}
```

- Per i dettagli sulle API, consulta la [CreateBucket](#) guida di riferimento all'API AWS SDK for Rust.

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

TRY.

```

    " determine our region from our session
    DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
    DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
    " When in the us-east-1 region, you must not specify a constraint
    " In all other regions, specify the region as the constraint
    IF lv_region = 'us-east-1'.
        CLEAR lo_constraint.
    ELSE.
        lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
    ENDIF.

    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
        io_createbucketconfiguration = lo_constraint ).
    MESSAGE 'S3 bucket created.' TYPE 'I'.
    CATCH /aws1/cx_s3_bucketalrddyexists.
        MESSAGE 'Bucket name already exists.' TYPE 'E'.
    CATCH /aws1/cx_s3_bktalrddyownedbyyou.
        MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
    ENDTRY.

```

- Per i dettagli sulle API, [CreateBucket](#) consulta AWS SDK for SAP ABAP API reference.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import AWSS3

public func createBucket(name: String) async throws {
    var input = CreateBucketInput(
        bucket: name
    )
}

```

```

        // For regions other than "us-east-1", you must set the
        locationConstraint in the createBucketConfiguration.
        // For more information, see LocationConstraint in the S3 API guide.
        // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_CreateBucket.html#API_CreateBucket_RequestBody
        if let region = configuration.region {
            if region != "us-east-1" {
                input.createBucketConfiguration =
S3ClientTypes.CreateBucketConfiguration(locationConstraint:
S3ClientTypes.BucketLocationConstraint(rawValue: region))
            }
        }

        do {
            _ = try await client.createBucket(input: input)
        }
        catch let error as BucketAlreadyOwnedByYou {
            print("The bucket '\(name)' already exists and is owned by you. You
may wish to ignore this exception.")
            throw error
        }
        catch {
            print("ERROR: ", dump(error, name: "Creating a bucket"))
            throw error
        }
    }
}

```

- Per i dettagli sull'API, consulta la [CreateBucket](#) guida di riferimento all'API AWS SDK for Swift.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **CreateMultiRegionAccessPoint** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare **CreateMultiRegionAccessPoint**.

Kotlin

SDK per Kotlin

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Configura il client di controllo S3 per inviare la richiesta alla Regione us-west-2.

```
suspend fun createS3ControlClient(): S3ControlClient {
    // Configure your S3ControlClient to send requests to US West
    (Oregon).
    val s3Control = S3ControlClient.fromEnvironment {
        region = "us-west-2"
    }
    return s3Control
}
```

Crea il punto di Multi-Region accesso.

```
suspend fun createMrap(
    s3Control: S3ControlClient,
    accountIdParam: String,
    bucketName1: String,
    bucketName2: String,
    mrapName: String,
): String {
    println("Creating MRAP ...")
    val createMrapResponse: CreateMultiRegionAccessPointResponse =
        s3Control.createMultiRegionAccessPoint {
            accountId = accountIdParam
            clientToken = UUID.randomUUID().toString()
            details {
                name = mrapName
                regions = listOf(
                    Region {
                        bucket = bucketName1
                    },
                ),
            }
        }
    return createMrapResponse.accessPointArn
}
```

```

        Region {
            bucket = bucketName2
        },
    )
}

val requestToken: String? = createMrapResponse.requestTokenArn

// Use the request token to check for the status of the
CreateMultiRegionAccessPoint operation.
if (requestToken != null) {
    waitForSucceededStatus(s3Control, requestToken, accountIdParam)
    println("MRAP created")
}

val getMrapResponse =
    s3Control.getMultiRegionAccessPoint(
        input = GetMultiRegionAccessPointRequest {
            accountId = accountIdParam
            name = mrapName
        },
    )
val mrapAlias = getMrapResponse.accessPoint?.alias
return "arn:aws:s3:::$accountIdParam:accesspoint/$mrpAlias"
}

```

Attendi che l' Multi-Region Access Point diventi disponibile.

```

suspend fun waitForSucceededStatus(
    s3Control: S3ControlClient,
    requestToken: String,
    accountIdParam: String,
    timeBetweenChecks: Duration = 1.minutes,
) {
    var describeResponse: DescribeMultiRegionAccessPointOperationResponse
    describeResponse = s3Control.describeMultiRegionAccessPointOperation(
        input = DescribeMultiRegionAccessPointOperationRequest {
            accountId = accountIdParam
            requestTokenArn = requestToken
        },
    )
}

```

```
var status: String? = describeResponse.asyncOperation?.requestStatus
while (status != "SUCCEEDED") {
    delay(timeBetweenChecks)
    describeResponse =
s3Control.describeMultiRegionAccessPointOperation(
    input = DescribeMultiRegionAccessPointOperationRequest {
        accountId = accountIdParam
        requestTokenArn = requestToken
    },
)
    status = describeResponse.asyncOperation?.requestStatus
    println(status)
}
}
```

- Per ulteriori informazioni, consulta la [Guida per gli sviluppatori di AWS SDK per Swift](#).
- Per i dettagli sull'API, consulta il riferimento [CreateMultiRegionAccessPoint](#) all'API AWS SDK for Kotlin.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **CreateMultipartUpload** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare CreateMultipartUpload.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- [Eseguire una copia in più parti](#)
- [Utilizzo dei checksum](#)
- [Utilizzare la funzionalità per l'integrità degli oggetti di Amazon S3](#)

C++

SDK per C++

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String
    &key,
                                Aws::S3::Model::ChecksumAlgorithm
checksumAlgorithm,
                                const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
        client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }
}
```

```
    return uploadID;
}
```

- Per i dettagli sull'API, consulta la [CreateMultipartUpload](#) sezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Il comando seguente completa un caricamento in più parti nel bucket `amzn-s3-demo-bucket` con la chiave `multipart/01`.

```
aws s3api create-multipart-upload --bucket amzn-s3-demo-bucket --key
'multipart/01'
```

Output:

```
{
  "Bucket": "amzn-s3-demo-bucket",
  "UploadId":
  "dfRtDYU0WWCCcH43C3WfbkR0NycyCpTJJvxu2i5GYkZ1jF.Yxwh6XG7WfS2vC4to6HiV6Yj1x.cph0gtNBtJ8P3
  "Key": "multipart/01"
}
```

Il nome del file completato sarà `01`, disponibile in una cartella denominata `multipart` nel bucket `amzn-s3-demo-bucket`. Salva l'ID di caricamento, la chiave e il nome del bucket da utilizzare con il comando `upload-part`.

- Per i dettagli sull'API, consulta [CreateMultipartUpload AWS CLI Command Reference](#).

Rust

SDK per Rust

Note

C'è altro da fare. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
// upload the file.
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await?;

let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
    "Missing upload_id after CreateMultipartUpload",
))?;
```

```
let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
    let upload_part_res = client
        .upload_part()
        .key(&key)
        .bucket(&bucket_name)
        .upload_id(upload_id)
        .body(stream)
        .part_number(part_number)
        .send()
        .await?;

    upload_parts.push(
```

```
        CompletedPart::builder()
            .e_tag(upload_part_res.e_tag.unwrap_or_default())
            .part_number(part_number)
            .build(),
    );
}
```

```
// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;
```

- Per i dettagli sulle API, consulta la [CreateMultipartUpload](#) guida di riferimento all'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **CreatePresignedPost** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `CreatePresignedPost`.

.NET

SDK per .NET (v4)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea un URL POST prefirmato.

```
/// <summary>
/// Create a presigned POST URL with conditions.
/// </summary>
/// <param name="s3Client">The Amazon S3 client.</param>
/// <param name="bucketName">The name of the bucket.</param>
/// <param name="objectKey">The object key (path) where the uploaded file
will be stored.</param>
/// <param name="expires">When the presigned URL expires.</param>
/// <param name="fields">Dictionary of fields to add to the form.</param>
/// <param name="conditions">List of conditions to apply.</param>
/// <returns>A CreatePresignedPostResponse object with URL and form fields.</
returns>
public async Task<CreatePresignedPostResponse> CreatePresignedPostAsync(
    IAmazonS3 s3Client,
    string bucketName,
    string objectKey,
    DateTime expires,
    Dictionary<string, string>? fields = null,
    List<S3PostCondition>? conditions = null)
{
    var request = new CreatePresignedPostRequest
    {
        BucketName = bucketName,
        Key = objectKey,
        Expires = expires
    };

    // Add custom fields if provided
    if (fields != null)
    {
        foreach (var field in fields)
```



```
        {
            request.Fields.Add(field.Key, field.Value);
        }
    }

    // Add conditions if provided
    if (conditions != null)
    {
        foreach (var condition in conditions)
        {
            request.Conditions.Add(condition);
        }
    }

    return await s3Client.CreatePresignedPostAsync(request);
}
```

- Per i dettagli sull'API, consulta la [CreatePresignedPost](#) sezione AWS SDK per .NET API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteBucket** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare DeleteBucket.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- [Informazioni di base](#)
- [Guida introduttiva alla gestione della configurazione](#)
- [Guida introduttiva agli store di funzionalità di apprendimento automatico](#)
- [Guida introduttiva allo storage di oggetti](#)

.NET

SDK per .NET (v4)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>
/// Shows how to delete an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket to delete.</
param>
/// <returns>A boolean value that represents the success or failure of
/// the delete operation.</returns>
public async Task<bool> DeleteBucketAsync(string bucketName)
{
    try
    {
        var request = new DeleteBucketRequest { BucketName = bucketName, };

        await _amazonS3.DeleteBucketAsync(request);
        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error deleting bucket: {ex.Message}");
        return false;
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteBucket](#) sezione AWS SDK per .NET API Reference.

Bash

AWS CLI con lo script Bash

 Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api delete-bucket \
        --bucket "$bucket_name")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
        return 1
    fi
}
```

```
    fi  
}
```

- Per i dettagli sull'API, consulta [DeleteBucket AWS CLI Command Reference](#).

C++

SDK per C++

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
bool AwsDoc::S3::deleteBucket(const Aws::String &bucketName,  
                               const Aws::S3::S3ClientConfiguration &clientConfig)  
{  
  
    Aws::S3::S3Client client(clientConfig);  
  
    Aws::S3::Model::DeleteBucketRequest request;  
    request.SetBucket(bucketName);  
  
    Aws::S3::Model::DeleteBucketOutcome outcome =  
        client.DeleteBucket(request);  
  
    if (!outcome.IsSuccess()) {  
        const Aws::S3::S3Error &err = outcome.GetError();  
        std::cerr << "Error: deleteBucket: " <<  
                    err.GetExceptionName() << ": " << err.GetMessage() <<  
std::endl;  
    } else {  
        std::cout << "The bucket was deleted" << std::endl;  
    }  
  
    return outcome.IsSuccess();  
}
```

- Per i dettagli sull'API, consulta la [DeleteBucket](#) sezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Il comando seguente elimina un bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api delete-bucket --bucket amzn-s3-demo-bucket --region us-east-1
```

- Per i dettagli sull'API, consulta [DeleteBucket AWS CLI](#) Command Reference.

Go

SDK per Go V2

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)
```

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// DeleteBucket deletes a bucket. The bucket must be empty or an error is
// returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
error {
    _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
        Bucket: aws.String(bucketName)})
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucketName)
            err = noBucket
        } else {
            log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
        }
    } else {
        err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
                bucketName)
        } else {
            log.Printf("Deleted %s.\n", bucketName)
        }
    }
    return err
}
```

- Per i dettagli sull'API, consulta la [DeleteBucket](#) sezione AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 * Deletes an S3 bucket asynchronously.
 *
 * @param bucket the name of the bucket to be deleted
 * @return a {@link CompletableFuture} that completes when the bucket
 * deletion is successful, or throws a {@link RuntimeException}
 * if an error occurs during the deletion process
 */
public CompletableFuture<Void> deleteBucketAsync(String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();

    CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
    response.whenComplete((deleteRes, ex) -> {
        if (deleteRes != null) {
            logger.info(bucket + " was deleted.");
        } else {
            throw new RuntimeException("An S3 exception occurred during
bucket deletion", ex);
        }
    });
    return response.thenApply(r -> null);
}
```

- Per i dettagli sull'API, consulta la [DeleteBucket](#) sezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina il bucket.

```
import {
  DeleteBucketCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Delete an Amazon S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new DeleteBucketCommand({
    Bucket: bucketName,
  });

  try {
    await client.send(command);
    console.log("Bucket was deleted.");
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting bucket. The bucket doesn't exist.`
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting the bucket. ${caught.name}:`
        + `${caught.message}`
      );
    }
  }
}
```



```
    );  
  } else {  
    throw caught;  
  }  
}  
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, consulta la [DeleteBucket](#) sezione AWS SDK per JavaScript API Reference.

PHP

SDK per PHP

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina un bucket vuoto.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);  
  
try {  
    $this->s3client->deleteBucket([  
        'Bucket' => $this->bucketName,  
    ]);  
    echo "Deleted bucket $this->bucketName.\n";  
} catch (Exception $exception) {  
    echo "Failed to delete $this->bucketName with error: " . $exception-  
>getMessage();  
    exit("Please fix error with bucket deletion before continuing.");  
}
```

- Per i dettagli sull'API, consulta la [DeleteBucket](#) sezione AWS SDK per PHP API Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando rimuove tutti gli oggetti e le relative versioni dal bucket 'test-files', quindi elimina il bucket. Il comando richiederà una conferma prima di procedere. Aggiungere il parametro `-Force` per disabilitare le richieste di conferma. Nota che i bucket non vuoti non possono essere eliminati.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- Per i dettagli sull'API, vedere [DeleteBucket](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando rimuove tutti gli oggetti e le relative versioni dal bucket 'test-files', quindi elimina il bucket. Il comando richiederà una conferma prima di procedere. Aggiungere il parametro `-Force` per disabilitare le richieste di conferma. Nota che i bucket non vuoti non possono essere eliminati.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- Per i dettagli sull'API, vedere [DeleteBucket](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Python

SDK per Python (Boto3)

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""
```

```

def __init__(self, bucket):
    """
    :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                    that wraps bucket actions in a class-like structure.
    """
    self.bucket = bucket
    self.name = bucket.name

def delete(self):
    """
    Delete the bucket. The bucket must be empty or an error is raised.
    """
    try:
        self.bucket.delete()
        self.bucket.wait_until_not_exists()
        logger.info("Bucket %s successfully deleted.", self.bucket.name)
    except ClientError:
        logger.exception("Couldn't delete bucket %s.", self.bucket.name)
        raise

```

- Per i dettagli sull'API, consulta [DeleteBucket AWS SDK for Python \(Boto3\) API Reference](#).

Ruby

SDK per Ruby

Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?
")

```

```

    answer = gets.chomp.downcase
    if answer == 'y'
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end

```

- Per i dettagli sull'API, consulta la [DeleteBucket](#) sezione AWS SDK per Ruby API Reference.

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

pub async fn delete_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<(), S3ExampleError> {
    let resp = client.delete_bucket().bucket(bucket_name).send().await;
    match resp {
        Ok(_) => Ok(()),
        Err(err) => {
            if err
                .as_service_error()
                .and_then(aws_sdk_s3::error::ProvideErrorMetadata::code)
                == Some("NoSuchBucket")
            {
                Ok(())
            } else {
                Err(S3ExampleError::from(err))
            }
        }
    }
}

```

```
}  
}  
}
```

- Per i dettagli sulle API, consulta la [DeleteBucket](#) guida di riferimento all'API AWS SDK for Rust.

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.
```

```
    lo_s3->deletebucket(  
        iv_bucket = iv_bucket_name ).  
    MESSAGE 'Deleted S3 bucket.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Per i dettagli sulle API, [DeleteBucket](#) consulta AWS SDK for SAP ABAP API reference.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import AWSS3

public func deleteBucket(name: String) async throws {
    let input = DeleteBucketInput(
        bucket: name
    )
    do {
        _ = try await client.deleteBucket(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Deleting a bucket"))
        throw error
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteBucket](#) guida di riferimento all'API AWS SDK for Swift.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteBucketAnalyticsConfiguration** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare DeleteBucketAnalyticsConfiguration.

CLI

AWS CLI

Come eliminare una configurazione di analisi per un bucket

L'esempio delete-bucket-analytics-configuration seguente rimuove la configurazione di analisi per il bucket e l'ID specificati.

```
aws s3api delete-bucket-analytics-configuration \
    --bucket amzn-s3-demo-bucket \
    --id 1
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [DeleteBucketAnalyticsConfiguration AWS CLI Command Reference](#).

PowerShell

Strumenti per PowerShell V4

Esempio 1: il comando rimuove il filtro di analisi denominato “testfilter” nel bucket S3 specificato.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AnalyticsId 'testfilter'
```

- Per i dettagli sull'API, vedere [DeleteBucketAnalyticsConfiguration](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: il comando rimuove il filtro di analisi denominato “testfilter” nel bucket S3 specificato.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AnalyticsId 'testfilter'
```

- Per i dettagli sull'API, vedere [DeleteBucketAnalyticsConfiguration](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteBucketCors** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare DeleteBucketCors.

.NET

SDK per .NET

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>
/// Deletes a CORS configuration from an Amazon S3 bucket.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used
/// to delete the CORS configuration from the bucket.</param>
private static async Task DeleteCORSConfigurationAsync(AmazonS3Client
client)
{
    DeleteCORSConfigurationRequest request = new
DeleteCORSConfigurationRequest()
    {
        BucketName = BucketName,
    };
    await client.DeleteCORSConfigurationAsync(request);
}
```

- Per i dettagli sull'API, consulta la [DeleteBucketCors](#) sezione AWS SDK per .NET API Reference.

CLI

AWS CLI

Il comando seguente elimina una configurazione di condivisione Cross-Origin delle risorse da un bucket denominato: `amzn-s3-demo-bucket`

```
aws s3api delete-bucket-cors --bucket amzn-s3-demo-bucket
```


- Per i dettagli sull'API, vedere [DeleteBucketCors](#) in AWS CLI Command Reference.

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete_cors(self):
        """
        Delete the CORS rules from the bucket.

        :param bucket_name: The name of the bucket to update.
        """
        try:
            self.bucket.Cors().delete()
            logger.info("Deleted CORS from bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete CORS from bucket '%s'.",
                             self.bucket.name)
            raise
```

- Per i dettagli sull'API, consulta [DeleteBucketCors AWS SDK for Python \(Boto3\) API Reference](#).

Ruby

SDK per Ruby

Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  # an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Deletes the CORS configuration of a bucket.
  #
  # @return [Boolean] True if the CORS rules were deleted; otherwise, false.
  def delete_cors
    @bucket_cors.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
    #{e.message}"
    false
  end
end
```

- Per i dettagli sull'API, consulta la [DeleteBucketCors](#) sezione AWS SDK per Ruby API Reference.

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.  
    lo_s3->deletebucketcors(  
        iv_bucket = iv_bucket_name ).  
    MESSAGE 'Bucket CORS configuration deleted.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Per i dettagli sulle API, [DeleteBucketCors](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteBucketEncryption** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare DeleteBucketEncryption.

CLI

AWS CLI

Come eliminare la configurazione di crittografia lato server di un bucket

L'esempio delete-bucket-encryption seguente elimina la configurazione di crittografia lato server del bucket specificato.

```
aws s3api delete-bucket-encryption \  
  --bucket amzn-s3-demo-bucket
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [DeleteBucketEncryption AWS CLI](#) Command Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando disabilita la crittografia abilitata per il bucket S3 specificato.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on  
target "s3casetestbucket".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

- Per i dettagli sull'API, vedere [DeleteBucketEncryption](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando disabilita la crittografia abilitata per il bucket S3 specificato.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on  
target "s3casetestbucket".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

- Per i dettagli sull'API, vedere [DeleteBucketEncryption](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteBucketInventoryConfiguration** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare DeleteBucketInventoryConfiguration.

CLI

AWS CLI

Come eliminare la configurazione di inventario di un bucket

L'esempio delete-bucket-inventory-configuration seguente elimina la configurazione dell'inventario con ID 1 per il bucket specificato.

```
aws s3api delete-bucket-inventory-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --id 1
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [DeleteBucketInventoryConfiguration AWS CLI](#) Command Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando rimuove l'inventario denominato «testInventoryName» corrispondente al bucket S3 specificato.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -  
InventoryId 'testInventoryName'
```

Output:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketInventoryConfiguration
(DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- Per i dettagli sull'API, vedere [DeleteBucketInventoryConfiguration](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando rimuove l'inventario denominato «testInventoryName» corrispondente al bucket S3 specificato.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -
InventoryId 'testInventoryName'
```

Output:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketInventoryConfiguration
(DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- Per i dettagli sull'API, vedere [DeleteBucketInventoryConfiguration](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteBucketLifecycle** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare DeleteBucketLifecycle.

.NET

SDK per .NET

Note

C'è altro su GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>
/// This method removes the Lifecycle configuration from the named
/// S3 bucket.
/// </summary>
/// <param name="client">The S3 client object used to call
/// the RemoveLifecycleConfigAsync method.</param>
/// <param name="bucketName">A string representing the name of the
/// S3 bucket from which the configuration will be removed.</param>
public static async Task RemoveLifecycleConfigAsync(IAmazonS3 client,
string bucketName)
{
    var request = new DeleteLifecycleConfigurationRequest()
    {
        BucketName = bucketName,
    };
    await client.DeleteLifecycleConfigurationAsync(request);
}
```

- Per i dettagli sull'API, consulta la [DeleteBucketLifecycle](#) sezione AWS SDK per .NET API Reference.

CLI

AWS CLI

Il comando seguente elimina una configurazione del ciclo di vita da un bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api delete-bucket-lifecycle --bucket amzn-s3-demo-bucket
```

- Per i dettagli sull'API, consulta [DeleteBucketLifecycle AWS CLI Command Reference](#).

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete_lifecycle_configuration(self):
        """
        Remove the lifecycle configuration from the specified bucket.
        """
        try:
            self.bucket.LifecycleConfiguration().delete()
            logger.info(
                "Deleted lifecycle configuration for bucket '%s'.",
                self.bucket.name
            )
        except ClientError:
            logger.exception(
                "Couldn't delete lifecycle configuration for bucket '%s'.",
                self.bucket.name,
```



```
)  
raise
```

- Per i dettagli sull'API, consulta [DeleteBucketLifecycle AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.  
  lo_s3->deletebucketlifecycle(  
    iv_bucket = iv_bucket_name ).  
  MESSAGE 'Bucket lifecycle configuration deleted.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
  MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Per i dettagli sulle API, [DeleteBucketLifecycle](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteBucketMetricsConfiguration** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare DeleteBucketMetricsConfiguration.

CLI

AWS CLI

Come eliminare una configurazione delle metriche per un bucket

L'esempio `delete-bucket-metrics-configuration` seguente rimuove la configurazione delle metriche per il bucket e l'ID specificati.

```
aws s3api delete-bucket-metrics-configuration \  
    --bucket amzn-s3-demo-bucket \  
    --id 123
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [DeleteBucketMetricsConfiguration AWS CLI Command Reference](#).

PowerShell

Strumenti per PowerShell V4

Esempio 1: il comando rimuove il filtro delle metriche denominato “testmetrics” nel bucket S3 specificato.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testmetrics'
```

- Per i dettagli sull'API, vedere [DeleteBucketMetricsConfiguration](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: il comando rimuove il filtro delle metriche denominato “testmetrics” nel bucket S3 specificato.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testmetrics'
```

- Per i dettagli sull'API, vedere [DeleteBucketMetricsConfiguration](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteBucketPolicy** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare DeleteBucketPolicy.

C++

SDK per C++

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
bool AwsDoc::S3::deleteBucketPolicy(const Aws::String &bucketName,
                                     const Aws::S3::S3ClientConfiguration
                                     &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
        client.DeleteBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucketPolicy: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
        std::endl;
    } else {
        std::cout << "Policy was deleted from the bucket." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, consulta la [DeleteBucketPolicy](#) sezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Il comando seguente elimina una policy di bucket denominata `amzn-s3-demo-bucket`.

```
aws s3api delete-bucket-policy --bucket amzn-s3-demo-bucket
```

- Per i dettagli sull'API, consulta [DeleteBucketPolicy AWS CLI](#) Command Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */

public class DeleteBucketPolicy {
    public static void main(String[] args) {
```

```

    final String usage = ""

        Usage:
            <bucketName>

        Where:
            bucketName - The Amazon S3 bucket to delete the policy from
(for example, bucket1).""";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    deleteS3BucketPolicy(s3, bucketName);
    s3.close();
}

/**
 * Deletes the S3 bucket policy for the specified bucket.
 *
 * @param s3 the {@link S3Client} instance to use for the operation
 * @param bucketName the name of the S3 bucket for which the policy should be
deleted
 *
 * @throws S3Exception if there is an error deleting the bucket policy
 */
public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
    DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketPolicy(delReq);
        System.out.println("Done!");
    }
}

```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteBucketPolicy](#) sezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina la policy del bucket.

```
import {
    DeleteBucketPolicyCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Remove the policy from an Amazon S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
    const client = new S3Client({});

    try {
        await client.send(
            new DeleteBucketPolicyCommand({
                Bucket: bucketName,
            }),
        );
    }
```

```
);
console.log(`Bucket policy deleted from "${bucketName}".`);
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "NoSuchBucket"
  ) {
    console.error(
      `Error from S3 while deleting policy from ${bucketName}. The bucket
doesn't exist.`,
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while deleting policy from ${bucketName}.  ${caught.name}:
${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, consulta la [DeleteBucketPolicy](#) sezione AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {
    val request =
        DeleteBucketPolicyRequest {
            bucket = bucketName
        }
}
```

```
    }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucketPolicy(request)
        println("Done!")
    }
}
```

- Per i dettagli sull'API, [DeleteBucketPolicy](#) consulta AWS SDK for Kotlin API reference.

PowerShell

Strumenti per V4 PowerShell

Esempio 1: il comando rimuove la policy di bucket associata al bucket S3 specificato.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [DeleteBucketPolicy](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: il comando rimuove la policy di bucket associata al bucket S3 specificato.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [DeleteBucketPolicy](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Python

SDK per Python (Boto3)

Note

C'è altro su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).


```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete_policy(self):
        """
        Delete the security policy from the bucket.
        """
        try:
            self.bucket.Policy().delete()
            logger.info("Deleted policy for bucket '%s'.", self.bucket.name)
        except ClientError:
            logger.exception(
                "Couldn't delete policy for bucket '%s'.", self.bucket.name
            )
            raise
```

- Per i dettagli sull'API, consulta [DeleteBucketPolicy AWS SDK for Python \(Boto3\) API Reference](#).

Ruby

SDK per Ruby

Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
  # configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  def delete_policy
    @bucket_policy.delete
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't delete the policy from #{@bucket_policy.bucket.name}. Here's
    why: #{e.message}"
    false
  end
end

end
```

- Per i dettagli sull'API, consulta la [DeleteBucketPolicy](#) sezione AWS SDK per Ruby API Reference.

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

TRY.

```
lo_s3->deletebucketpolicy(
  iv_bucket = iv_bucket_name ).
MESSAGE 'Bucket policy deleted.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
MESSAGE 'Bucket does not exist.' TYPE 'E'.
```

```
ENDTRY.
```

- Per i dettagli sulle API, [DeleteBucketPolicy](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteBucketReplication** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare DeleteBucketReplication.

CLI

AWS CLI

Il comando seguente elimina una configurazione di replica da un bucket denominato amzn-s3-demo-bucket.

```
aws s3api delete-bucket-replication --bucket amzn-s3-demo-bucket
```

- Per i dettagli sull'API, consulta [DeleteBucketReplication AWS CLI](#) Command Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: elimina la configurazione di replica associata al bucket denominato "amzn-s3-demo-bucket". Nota che questa operazione richiede l'autorizzazione per l'azione s3:DeleteReplicationConfiguration . Prima di procedere con l'operazione, viene richiesta una conferma. Per eliminare la conferma, utilizza l'opzione -Force.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Per i dettagli sull'API, vedere [DeleteBucketReplication](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: elimina la configurazione di replica associata al bucket denominato “amzn-s3-demo-bucket”. Nota che questa operazione richiede l'autorizzazione per l'azione `s3:DeleteReplicationConfiguration`. Prima di procedere con l'operazione, viene richiesta una conferma. Per eliminare la conferma, utilizza l'opzione `-Force`.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Per i dettagli sull'API, vedere [DeleteBucketReplication](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteBucketTagging** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare DeleteBucketTagging.

CLI

AWS CLI

Il comando seguente elimina una configurazione di tagging da un bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api delete-bucket-tagging --bucket amzn-s3-demo-bucket
```

- Per i dettagli sull'API, consulta [DeleteBucketTagging AWS CLI](#) Command Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando rimuove tutti i tag associati al bucket S3 specificato.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
"amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- Per i dettagli sull'API, vedere [DeleteBucketTagging](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando rimuove tutti i tag associati al bucket S3 specificato.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
"amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- Per i dettagli sull'API, vedere [DeleteBucketTagging](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteBucketWebsite** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare DeleteBucketWebsite.

C++

SDK per C++

 Note

C'è altro su GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
bool AwsDoc::S3::deleteBucketWebsite(const Aws::String &bucketName,
                                       const Aws::S3::S3ClientConfiguration
                                       &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
        client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteBucketWebsite: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Website configuration was removed." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, consulta la [DeleteBucketWebsite](#) sezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Il comando seguente elimina una configurazione di siti web da un bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api delete-bucket-website --bucket amzn-s3-demo-bucket
```

- Per i dettagli sull'API, consulta [DeleteBucketWebsite AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */

public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""
```

```
Usage:      <bucketName>

Where:
    bucketName - The Amazon S3 bucket to delete the website
configuration from.
""";

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String bucketName = args[0];
System.out.format("Deleting website configuration for Amazon S3 bucket:
%s\n", bucketName);
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

deleteBucketWebsiteConfig(s3, bucketName);
System.out.println("Done!");
s3.close();
}

/**
 * Deletes the website configuration for an Amazon S3 bucket.
 *
 * @param s3 The {@link S3Client} instance used to interact with Amazon S3.
 * @param bucketName The name of the S3 bucket for which the website
configuration should be deleted.
 * @throws S3Exception If an error occurs while deleting the website
configuration.
 */
public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName)
{
    DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketWebsite(delReq);
    } catch (S3Exception e) {
```



```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.out.println("Failed to delete website configuration!");
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteBucketWebsite](#) sezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina la configurazione del sito web dal bucket.

```
import {
    DeleteBucketWebsiteCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Remove the website configuration for a bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
    const client = new S3Client({});

    try {
        await client.send(
            new DeleteBucketWebsiteCommand({
                Bucket: bucketName,
            }),
        );
    }
```

```
// The response code will be successful for both removed configurations and
// configurations that did not exist in the first place.
console.log(
    `The bucket "${bucketName}" is not longer configured as a website, or it
never was.` ,
);
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while removing website configuration from ${bucketName}.
The bucket doesn't exist.` ,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while removing website configuration from ${bucketName}.
${caught.name}: ${caught.message}` ,
        );
    } else {
        throw caught;
    }
}
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, consulta la [DeleteBucketWebsite](#) sezione AWS SDK per JavaScript API Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando disabilita la proprietà di hosting di siti web statici del bucket S3 specificato.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
"amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- Per i dettagli sull'API, vedere [DeleteBucketWebsite](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando disabilita la proprietà di hosting di siti web statici del bucket S3 specificato.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
"amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- Per i dettagli sull'API, vedere [DeleteBucketWebsite](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteObject** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare DeleteObject.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- [Utilizzare la funzionalità per l'integrità degli oggetti di Amazon S3](#)

- [Utilizzo degli oggetti con versione](#)

.NET

SDK per .NET

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina un oggetto in un bucket S3 senza controllo delle versioni.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete an object from a non-versioned Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DeleteObject
{
    /// <summary>
    /// The Main method initializes the necessary variables and then calls
    /// the DeleteObjectNonVersionedBucketAsync method to delete the object
    /// named by the keyName parameter.
    /// </summary>
    public static async Task Main()
    {
        const string bucketName = "amzn-s3-demo-bucket";
        const string keyName = "testfile.txt";

        // If the Amazon S3 bucket is located in an AWS Region other than the
        // Region of the default account, define the AWS Region for the
        // Amazon S3 bucket in your call to the AmazonS3Client constructor.
        // For example RegionEndpoint.USWest2.
        IAmazonS3 client = new AmazonS3Client();
        await DeleteObjectNonVersionedBucketAsync(client, bucketName,
keyName);
```

```

    }

    /// <summary>
    /// The DeleteObjectNonVersionedBucketAsync takes care of deleting the
    /// desired object from the named bucket.
    /// </summary>
    /// <param name="client">An initialized Amazon S3 client used to delete
    /// an object from an Amazon S3 bucket.</param>
    /// <param name="bucketName">The name of the bucket from which the
    /// object will be deleted.</param>
    /// <param name="keyName">The name of the object to delete.</param>
    public static async Task DeleteObjectNonVersionedBucketAsync(IAmazonS3
client, string bucketName, string keyName)
    {
        try
        {
            var deleteObjectRequest = new DeleteObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
            };

            Console.WriteLine($"Deleting object: {keyName}");
            await client.DeleteObjectAsync(deleteObjectRequest);
            Console.WriteLine($"Object: {keyName} deleted from
{bucketName}.");
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error encountered on server.
Message: '{ex.Message}' when deleting an object.");
        }
    }
}

```

Elimina un oggetto in un bucket S3 con controllo delle versioni.

```

using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

```

```

/// <summary>
/// This example creates an object in an Amazon Simple Storage Service
/// (Amazon S3) bucket and then deletes the object version that was
/// created.
/// </summary>
public class DeleteObjectVersion
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "versioned-object.txt";

        // If the AWS Region of the default user is different from the AWS
        // Region of the Amazon S3 bucket, pass the AWS Region of the
        // bucket region to the Amazon S3 client object's constructor.
        // Define it like this:
        //      RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        IAmazonS3 client = new AmazonS3Client();

        await CreateAndDeleteObjectVersionAsync(client, bucketName, keyName);
    }

    /// <summary>
    /// This method creates and then deletes a versioned object.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
    /// create and delete the object.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
    /// object will be created and deleted.</param>
    /// <param name="keyName">The key name of the object to create.</param>
    public static async Task CreateAndDeleteObjectVersionAsync(IAmazonS3
client, string bucketName, string keyName)
    {
        try
        {
            // Add a sample object.
            string versionID = await PutAnObject(client, bucketName,
keyName);

            // Delete the object by specifying an object key and a version
ID.

            DeleteObjectRequest request = new DeleteObjectRequest()
            {

```

```

        BucketName = bucketName,
        Key = keyName,
        VersionId = versionID,
    };

    Console.WriteLine("Deleting an object");
    await client.DeleteObjectAsync(request);
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error: {ex.Message}");
}
}

/// <summary>
/// This method is used to create the temporary Amazon S3 object.
/// </summary>
/// <param name="client">The initialized Amazon S3 object which will be
used
/// to create the temporary Amazon S3 object.</param>
/// <param name="bucketName">The name of the Amazon S3 bucket where the
object
/// will be created.</param>
/// <param name="objectKey">The name of the Amazon S3 object to create.</
param>
/// <returns>The Version ID of the created object.</returns>
public static async Task<string> PutAnObject(IAmazonS3 client, string
bucketName, string objectKey)
{
    PutObjectRequest request = new PutObjectRequest()
    {
        BucketName = bucketName,
        Key = objectKey,
        ContentBody = "This is the content body!",
    };

    PutObjectResponse response = await client.PutObjectAsync(request);
    return response.VersionId;
}
}

```

- Per i dettagli sull'API, consulta la [DeleteObject](#) sezione AWS SDK per .NET API Reference.

Bash

AWS CLI con lo script Bash

 Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_item_in_bucket
#
# This function deletes the specified file from the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.
#     $2 - The key (file name) in the bucket to delete.

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_item_in_bucket() {
    local bucket_name=$1
    local key=$2
    local response

    response=$(aws s3api delete-object \
        --bucket "$bucket_name" \
        --key "$key")

    # shellcheck disable=SC2181
```



```

if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api delete-object operation failed.\n
$response"
    return 1
fi
}

```

- Per i dettagli sull'API, consulta [DeleteObject AWS CLI Command Reference](#).

C++

SDK per C++

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

bool AwsDoc::S3::deleteObject(const Aws::String &objectKey,
                               const Aws::String &fromBucket,
                               const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
           .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
        std::endl;
    } else {
        std::cout << "Successfully deleted the object." << std::endl;
    }
}

```

```
    return outcome.IsSuccess();  
}
```

- Per i dettagli sull'API, consulta la [DeleteObject](#) sezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Il comando seguente elimina un oggetto denominato `test.txt` da un bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api delete-object --bucket amzn-s3-demo-bucket --key test.txt
```

Se il controllo delle versioni del bucket è abilitato, l'output conterrà l'ID della versione del contrassegno di eliminazione.

```
{  
  "VersionId": "9_gKg5vG56F.TTEUdwkxGpJ3tND1w1Gq",  
  "DeleteMarker": true  
}
```

Per ulteriori informazioni sull'eliminazione degli oggetti, consulta “Eliminazione di oggetti” nella Guida per gli sviluppatori di Amazon S3.

- Per i dettagli sull'API, consulta [DeleteObject AWS CLI](#) Command Reference.

Go

SDK per Go V2

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (
```

```
"bytes"
"context"
"errors"
"fmt"
"log"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
    string, versionId string, bypassGovernance bool) (bool, error) {
    deleted := false
    input := &s3.DeleteObjectInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    _, err := actor.S3Client.DeleteObject(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in %s.\n", key, bucket)
            err = noKey
        } else if errors.As(err, &apiErr) {
```

```
switch apiErr.ErrorCode() {
case "AccessDenied":
    log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
    err = nil
case "InvalidArgument":
    if bypassGovernance {
        log.Printf("You cannot specify bypass governance on a bucket without lock
enabled.")
        err = nil
    }
}
} else {
    err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
        ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
        time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.
\n", key, bucket)
    } else {
        deleted = true
    }
}
return deleted, err
}
```

- Per i dettagli sull'API, consulta la [DeleteObject](#) sezione AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

/**

```

    * Deletes an object from an S3 bucket asynchronously.
    *
    * @param bucketName the name of the S3 bucket
    * @param key         the key (file name) of the object to be deleted
    * @return a {@link CompletableFuture} that completes when the object has
    been deleted
    */
    public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
        DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
        response.whenComplete((deleteRes, ex) -> {
            if (deleteRes != null) {
                logger.info(key + " was deleted");
            } else {
                throw new RuntimeException("An S3 exception occurred during
delete", ex);
            }
        });

        return response.thenApply(r -> null);
    }

```

- Per i dettagli sull'API, consulta la [DeleteObject](#) sezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina un oggetto.

```
import {
  DeleteObjectCommand,
  S3Client,
  S3ServiceException,
  waitUntilObjectNotExists,
} from "@aws-sdk/client-s3";

/**
 * Delete one object from an Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteObjectCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    await waitUntilObjectNotExists(
      { client },
      { Bucket: bucketName, Key: key },
    );
    // A successful delete, or a delete for a non-existent object, both return
    // a 204 response code.
    console.log(
      `The object "${key}" from bucket "${bucketName}" was deleted, or it didn't
      exist.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting object from ${bucketName}. The bucket
        doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
```

```
        `Error from S3 while deleting object from ${bucketName}. ${caught.name}:  
        ${caught.message}`,  
        );  
    } else {  
        throw caught;  
    }  
}  
};
```

- Per i dettagli sull'API, consulta la [DeleteObject](#) sezione AWS SDK per JavaScript API Reference.

PHP

SDK per PHP

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public function deleteObject(string $bucketName, string $fileName, array  
$args = [])  
{  
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],  
$args);  
    try {  
        $this->client->deleteObject($parameters);  
        if ($this->verbose) {  
            echo "Deleted the object named: $fileName from $bucketName.\n";  
        }  
    } catch (AwsException $exception) {  
        if ($this->verbose) {  
            echo "Failed to delete $fileName from $bucketName with error:  
{ $exception->getMessage() }\n";  
            echo "Please fix error with object deletion before continuing."  
        }  
        throw $exception;  
    }  
}
```

```
}  
}
```

- Per i dettagli sull'API, [DeleteObject](#) consulta AWS SDK per PHP API Reference.

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina un oggetto.

```
class ObjectWrapper:  
    """Encapsulates S3 object actions."""  
  
    def __init__(self, s3_object):  
        """  
        :param s3_object: A Boto3 Object resource. This is a high-level resource  
in Boto3  
        that wraps object actions in a class-like structure.  
        """  
        self.object = s3_object  
        self.key = self.object.key  
  
    def delete(self):  
        """  
        Deletes the object.  
        """  
        try:  
            self.object.delete()  
            self.object.wait_until_not_exists()  
            logger.info(  
                "Deleted object '%s' from bucket '%s'.",  
                self.object.key,
```



```

        self.object.bucket_name,
    )
except ClientError:
    logger.exception(
        "Couldn't delete object '%s' from bucket '%s'.",
        self.object.key,
        self.object.bucket_name,
    )
    raise

```

Ripristina la versione precedente di un oggetto eliminando quelle successive.

```

def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

```

```

if version_id in [ver.version_id for ver in versions]:
    print(f"Rolling back to version {version_id}")
    for version in versions:
        if version.version_id != version_id:
            version.delete()
            print(f"Deleted version {version.version_id}")
        else:
            break

    print(f"Active version is now {bucket.Object(object_key).version_id}")
else:
    raise KeyError(
        f"{version_id} was not found in the list of versions for "
        f"{object_key}."
    )

```

Riattiva un oggetto eliminato rimuovendo il contrassegno di eliminazione attivo dell'oggetto.

```

def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
    marker.
    By removing the delete marker, we make the previous version the latest
    version
    and the object then presents as not deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

```

```

if "DeleteMarkers" in response:
    latest_version = response["DeleteMarkers"][0]
    if latest_version["IsLatest"]:
        logger.info(
            "Object %s was indeed deleted on %s. Let's revive it.",
            object_key,
            latest_version["LastModified"],
        )
        obj = bucket.Object(object_key)
        obj.Version(latest_version["VersionId"]).delete()
        logger.info(
            "Revived %s, active version is now %s with body '%s'",
            object_key,
            obj.version_id,
            obj.get()["Body"].read(),
        )
    else:
        logger.warning(
            "Delete marker is not the latest version for %s!", object_key
        )
elif "Versions" in response:
    logger.warning("Got an active version for %s, nothing to do.",
object_key)
else:
    logger.error("Couldn't get any version info for %s.", object_key)

```

Crea un gestore Lambda che rimuove un contrassegno di eliminazione da un oggetto S3. Questo gestore può essere utilizzato per ripulire in modo efficiente i contrassegni di eliminazione estranei di un bucket con versione.

```

import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")

```

```

def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of
    the
               operation. When the result code is TemporaryFailure, S3 retries the
               operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]

    try:
        obj_key = parse.unquote_plus(task["s3Key"], encoding="utf-8")
        obj_version_id = task["s3VersionId"]
        bucket_name = task["s3BucketArn"].split(":")[-1]

        logger.info(
            "Got task: remove delete marker %s from object %s.", obj_version_id,
            obj_key
        )

        try:
            # If this call does not raise an error, the object version is not a
            delete
            # marker and should not be deleted.
            response = s3.head_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
            )
            result_code = "PermanentFailure"
            result_string = (

```

```

        f"Object {obj_key}, ID {obj_version_id} is not " f"a delete
marker."
    )

    logger.debug(response)
    logger.warning(result_string)
except ClientError as error:
    delete_marker = error.response["ResponseMetadata"]
["HTTPHeaders"].get(
        "x-amz-delete-marker", "false"
    )
    if delete_marker == "true":
        logger.info(
            "Object %s, version %s is a delete marker.", obj_key,
obj_version_id
        )
        try:
            s3.delete_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
            )
            result_code = "Succeeded"
            result_string = (
                f"Successfully removed delete marker "
                f"{obj_version_id} from object {obj_key}."
            )
            logger.info(result_string)
        except ClientError as error:
            # Mark request timeout as a temporary failure so it will be
retried.

            if error.response["Error"]["Code"] == "RequestTimeout":
                result_code = "TemporaryFailure"
                result_string = (
                    f"Attempt to remove delete marker from "
                    f"object {obj_key} timed out."
                )
                logger.info(result_string)
            else:
                raise
    else:
        raise ValueError(
            f"The x-amz-delete-marker header is either not "
            f"present or is not 'true'."
        )
except Exception as error:

```

```

        # Mark all other exceptions as permanent failures.
        result_code = "PermanentFailure"
        result_string = str(error)
        logger.exception(error)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
            }
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }

```

- Per i dettagli sull'API, consulta [DeleteObject AWS SDK for Python \(Boto3\) API Reference](#).

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

/// Delete an object from a bucket.
pub async fn remove_object(
    client: &aws_sdk_s3::Client,
    bucket: &str,
    key: &str,
) -> Result<(), S3ExampleError> {
    client
        .delete_object()

```

```
.bucket(bucket)
.key(key)
.send()
.await?;

// There are no modeled errors to handle when deleting an object.

Ok(())
}
```

- Per i dettagli sulle API, consulta la [DeleteObject](#) guida di riferimento all'API AWS SDK for Rust.

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.
  lo_s3->deleteobject(
    iv_bucket = iv_bucket_name
    iv_key = iv_object_key ).
  MESSAGE 'Object deleted from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Per i dettagli sulle API, [DeleteObject](#) consulta AWS SDK for SAP ABAP API reference.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import AWSS3

public func deleteFile(bucket: String, key: String) async throws {
    let input = DeleteObjectInput(
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.deleteObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Deleting a file. "))
        throw error
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteObject](#) guida di riferimento all'API AWS SDK for Swift.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteObjectTagging** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare DeleteObjectTagging.

CLI

AWS CLI

Come eliminare i set di tag di un oggetto

L'esempio `delete-object-tagging` seguente elimina il tag con la chiave specificata dall'oggetto `doc1.rtf`.

```
aws s3api delete-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc1.rtf
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [DeleteObjectTagging AWS CLI Command Reference](#).

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando rimuove tutti i tag associati all'oggetto con la chiave "testfile.txt" nel bucket S3 specificato.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -  
Select '^Key'
```

Output:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target  
"testfile.txt".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y  
testfile.txt
```

- Per i dettagli sull'API, vedere [DeleteObjectTagging](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando rimuove tutti i tag associati all'oggetto con la chiave "testfile.txt" nel bucket S3 specificato.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -  
Select '^Key'
```

Output:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target  
"testfile.txt".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y  
testfile.txt
```

- Per i dettagli sull'API, vedere [DeleteObjectTagging](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteObjects** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare DeleteObjects.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- [Informazioni di base](#)
- [Eliminare tutti gli oggetti in un bucket](#)
- [Guida introduttiva allo storage di oggetti](#)

.NET

SDK per .NET (v4)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>
/// Delete all of the objects stored in an existing Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket from which the
/// contents will be deleted.</param>
/// <returns>A boolean value that represents the success or failure of
/// deleting all of the objects in the bucket.</returns>
public async Task<bool> DeleteBucketContentsAsync(string bucketName)
{
    // Iterate over the contents of the bucket and delete all objects.
    try
    {
        // Delete all objects in the bucket.
        var deleteList = await ListBucketContentsAsync(bucketName, false);
        if (deleteList != null && deleteList.Any())
        {
            await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
            {
                BucketName = bucketName,
                Objects = deleteList.Select(o => new KeyVersion { Key =
o.Key }).ToList(),
            });
        }

        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error deleting objects: {ex.Message}");
        return false;
    }
}
```

```
}
```

- Per i dettagli sull'API, [DeleteObjects](#) consulta AWS SDK per .NET API Reference.

SDK per .NET

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Eliminare più oggetti da un bucket S3 senza controllo delle versioni.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete multiple objects from an Amazon Simple
/// Storage Service (Amazon S3) bucket.
/// </summary>
public class DeleteMultipleObjects
{
    /// <summary>
    /// The Main method initializes the Amazon S3 client and the name of
    /// the bucket and then passes those values to MultiObjectDeleteAsync.
    /// </summary>
    public static async Task Main()
    {
        const string bucketName = "amzn-s3-demo-bucket";

        // If the Amazon S3 bucket from which you wish to delete objects is
        // located in the same AWS Region as the default user, define the
        // AWS Region for the Amazon S3 bucket as a parameter to the client
        // constructor.
        IAmazonS3 s3Client = new AmazonS3Client();
```

not

```

        await MultiObjectDeleteAsync(s3Client, bucketName);
    }

    /// <summary>
    /// This method uses the passed Amazon S3 client to first create and then
    /// delete three files from the named bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// Amazon S3 methods.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket where
objects
    /// will be created and then deleted.</param>
    public static async Task MultiObjectDeleteAsync(IAmazonS3 client, string
bucketName)
    {
        // Create three sample objects which we will then delete.
        var keysAndVersions = await PutObjectsAsync(client, 3, bucketName);

        // Now perform the multi-object delete, passing the key names and
        // version IDs. Since we are working with a non-versioned bucket,
        // the object keys collection includes null version IDs.
        DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest
        {
            BucketName = bucketName,
            Objects = keysAndVersions,
        };

        // You can add a specific object key to the delete request using the
        // AddKey method of the multiObjectDeleteRequest.
        try
        {
            DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
            Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
        }
        catch (DeleteObjectsException e)
        {
            PrintDeletionErrorStatus(e);
        }
    }

```

```

    /// <summary>
    /// Prints the list of errors raised by the call to DeleteObjectsAsync.
    /// </summary>
    /// <param name="ex">A collection of exceptions returned by the call to
    /// DeleteObjectsAsync.</param>
    public static void PrintDeletionErrorStatus(DeleteObjectsException ex)
    {
        DeleteObjectsResponse errorResponse = ex.Response;
        Console.WriteLine("x {0}", errorResponse.DeletedObjects.Count);

        Console.WriteLine($"Successfully deleted
{errorResponse.DeletedObjects.Count}.");
        Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");

        Console.WriteLine("Printing error data...");
        foreach (DeleteError deleteError in errorResponse.DeleteErrors)
        {
            Console.WriteLine($"Object Key:
{deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
        }
    }

    /// <summary>
    /// This method creates simple text file objects that can be used in
    /// the delete method.
    /// </summary>
    /// <param name="client">The Amazon S3 client used to call
    PutObjectAsync.</param>
    /// <param name="number">The number of objects to create.</param>
    /// <param name="bucketName">The name of the bucket where the objects
    /// will be created.</param>
    /// <returns>A list of keys (object keys) and versions that the calling
    /// method will use to delete the newly created files.</returns>
    public static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
    client, int number, string bucketName)
    {
        List<KeyVersion> keys = new List<KeyVersion>();
        for (int i = 0; i < number; i++)
        {
            string key = "ExampleObject-" + new System.Random().Next();
            PutObjectRequest request = new PutObjectRequest
            {
                BucketName = bucketName,

```

```

        Key = key,
        ContentBody = "This is the content body!",
    };

    PutObjectResponse response = await
client.PutObjectAsync(request);

    // For non-versioned bucket operations, we only need the
    // object key.
    KeyVersion keyVersion = new KeyVersion
    {
        Key = key,
    };
    keys.Add(keyVersion);
}

return keys;
}
}

```

Eliminare più oggetti da un bucket S3 con controllo delle versioni.

```

using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete objects in a version-enabled Amazon
/// Simple StorageService (Amazon S3) bucket.
/// </summary>
public class DeleteMultipleObjects
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";

        // If the AWS Region for your Amazon S3 bucket is different from
        // the AWS Region of the default user, define the AWS Region for
        // the Amazon S3 bucket and pass it to the client constructor
    }
}

```

```

        // like this:
        // RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        IAmazonS3 s3Client;

        s3Client = new AmazonS3Client();
        await DeleteMultipleObjectsFromVersionedBucketAsync(s3Client,
bucketName);
    }

    /// <summary>
    /// This method removes multiple versions and objects from a
    /// version-enabled Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
    /// RemoveDeleteMarkersAsync.</param>
    /// <param name="bucketName">The name of the bucket from which to delete
    /// objects.</param>
    public static async Task
DeleteMultipleObjectsFromVersionedBucketAsync(IAmazonS3 client, string
bucketName)
    {
        // Delete objects (specifying object version in the request).
        await DeleteObjectVersionsAsync(client, bucketName);

        // Delete objects (without specifying object version in the request).
        var deletedObjects = await DeleteObjectsAsync(client, bucketName);

        // Additional exercise - remove the delete markers Amazon S3 returned
from
        // the preceding response. This results in the objects reappearing
        // in the bucket (you can verify the appearance/disappearance of
        // objects in the console).
        await RemoveDeleteMarkersAsync(client, bucketName, deletedObjects);
    }

    /// <summary>
    /// Creates and then deletes non-versioned Amazon S3 objects and then
deletes
    /// them again. The method returns a list of the Amazon S3 objects
deleted.
    /// </summary>

```



```

call
    /// <param name="client">The initialized Amazon S3 client object used to
    /// PubObjectsAsync and NonVersionedDeleteAsync.</param>
    /// <param name="bucketName">The name of the bucket where the objects
    /// will be created and then deleted.</param>
    /// <returns>A list of DeletedObjects.</returns>
    public static async Task<List<DeletedObject>>
DeleteObjectsAsync(IAmazonS3 client, string bucketName)
    {
        // Upload the sample objects.
        var keysAndVersions2 = await PutObjectsAsync(client, bucketName, 3);

        // Delete objects using only keys. Amazon S3 creates a delete marker
        // returns its version ID in the response.
        List<DeletedObject> deletedObjects = await
NonVersionedDeleteAsync(client, bucketName, keysAndVersions2);
        return deletedObjects;
    }

    /// <summary>
    /// This method creates several temporary objects and then deletes them.
    /// </summary>
    /// <param name="client">The S3 client.</param>
    /// <param name="bucketName">Name of the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task DeleteObjectVersionsAsync(IAmazonS3 client,
string bucketName)
    {
        // Upload the sample objects.
        var keysAndVersions1 = await PutObjectsAsync(client, bucketName, 3);

        // Delete the specific object versions.
        await VersionedDeleteAsync(client, bucketName, keysAndVersions1);
    }

    /// <summary>
    /// Displays the list of information about deleted files to the console.
    /// </summary>
    /// <param name="e">Error information from the delete process.</param>
    private static void DisplayDeletionErrors(DeleteObjectsException e)
    {
        var errorResponse = e.Response;

```

```

        Console.WriteLine($"No. of objects successfully deleted =
{errorResponse.DeletedObjects.Count}");
        Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");
        Console.WriteLine("Printing error data...");
        foreach (var deleteError in errorResponse.DeleteErrors)
        {
            Console.WriteLine($"Object Key:
{deleteError.Key}\\t{deleteError.Code}\\t{deleteError.Message}");
        }
    }

    /// <summary>
    /// Delete multiple objects from a version-enabled bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
    /// RemoveDeleteMarkersAsync.</param>
    /// <param name="bucketName">The name of the bucket from which to delete
    /// objects.</param>
    /// <param name="keys">A list of key names for the objects to delete.</
param>
    private static async Task VersionedDeleteAsync(IAmazonS3 client, string
bucketName, List<KeyVersion> keys)
    {
        var multiObjectDeleteRequest = new DeleteObjectsRequest
        {
            BucketName = bucketName,
            Objects = keys, // This includes the object keys and specific
version IDs.
        };

        try
        {
            Console.WriteLine("Executing VersionedDelete...");
            DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
            Console.WriteLine($"Successfully deleted all the
{response.DeletedObjects.Count} items");
        }
        catch (DeleteObjectsException ex)
        {
            DisplayDeletionErrors(ex);
        }
    }

```

```

    }
}

/// <summary>
/// Deletes multiple objects from a non-versioned Amazon S3 bucket.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
call
/// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
/// RemoveDeleteMarkersAsync.</param>
/// <param name="bucketName">The name of the bucket from which to delete
/// objects.</param>
/// <param name="keys">A list of key names for the objects to delete.</
param>
/// <returns>A list of the deleted objects.</returns>
private static async Task<List<DeletedObject>>
NonVersionedDeleteAsync(IAmazonS3 client, string bucketName, List<KeyVersion>
keys)
{
    // Create a request that includes only the object key names.
    DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest();
    multiObjectDeleteRequest.BucketName = bucketName;

    foreach (var key in keys)
    {
        multiObjectDeleteRequest.AddKey(key.Key);
    }

    // Execute DeleteObjectsAsync.
    // The DeleteObjectsAsync method adds a delete marker for each
    // object deleted. You can verify that the objects were removed
    // using the Amazon S3 console.
    DeleteObjectsResponse response;
    try
    {
        Console.WriteLine("Executing NonVersionedDelete...");
        response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
        Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
    }
    catch (DeleteObjectsException ex)
    {

```

```

        DisplayDeletionErrors(ex);
        throw; // Some deletions failed. Investigate before continuing.
    }

    // This response contains the DeletedObjects list which we use to
delete the delete markers.
    return response.DeletedObjects;
}

/// <summary>
/// Deletes the markers left after deleting the temporary objects.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
call
/// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
/// RemoveDeleteMarkersAsync.</param>
/// <param name="bucketName">The name of the bucket from which to delete
/// objects.</param>
/// <param name="deletedObjects">A list of the objects that were
deleted.</param>
private static async Task RemoveDeleteMarkersAsync(IAmazonS3 client,
string bucketName, List<DeletedObject> deletedObjects)
{
    var keyVersionList = new List<KeyVersion>();

    foreach (var deletedObject in deletedObjects)
    {
        KeyVersion keyVersion = new KeyVersion
        {
            Key = deletedObject.Key,
            VersionId = deletedObject.DeleteMarkerVersionId,
        };
        keyVersionList.Add(keyVersion);
    }

    // Create another request to delete the delete markers.
    var multiObjectDeleteRequest = new DeleteObjectsRequest
    {
        BucketName = bucketName,
        Objects = keyVersionList,
    };

    // Now, delete the delete marker to bring your objects back to the
bucket.

```

```

        try
        {
            Console.WriteLine("Removing the delete markers .....");
            var deleteObjectResponse = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
            Console.WriteLine($"Successfully deleted the
{deleteObjectResponse.DeletedObjects.Count} delete markers");
        }
        catch (DeleteObjectsException ex)
        {
            DisplayDeletionErrors(ex);
        }
    }

    /// <summary>
    /// Create temporary Amazon S3 objects to show how object deletion works
in an
    /// Amazon S3 bucket with versioning enabled.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// PutObjectAsync to create temporary objects for the example.</param>
    /// <param name="bucketName">A string representing the name of the S3
    /// bucket where we will create the temporary objects.</param>
    /// <param name="number">The number of temporary objects to create.</
param>
    /// <returns>A list of the KeyVersion objects.</returns>
    private static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
client, string bucketName, int number)
    {
        var keys = new List<KeyVersion>();

        for (var i = 0; i < number; i++)
        {
            string key = "ObjectToDelete-" + new System.Random().Next();
            PutObjectRequest request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = key,
                ContentBody = "This is the content body!",
            };

            var response = await client.PutObjectAsync(request);
            KeyVersion keyVersion = new KeyVersion

```

```

        {
            Key = key,
            VersionId = response.VersionId,
        };

        keys.Add(keyVersion);
    }

    return keys;
}
}

```

- Per i dettagli sull'API, [DeleteObjects](#) consulta AWS SDK per .NET API Reference.

Bash

AWS CLI con lo script Bash

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.

```

```
#      $2 - A list of keys in the bucket to delete.

# Returns:
#      0 - If successful.
#      1 - If it fails.
#####
function delete_items_in_bucket() {
    local bucket_name=$1
    local keys=$2
    local response

    # Create the JSON for the items to delete.
    local delete_items
    delete_items="{\"Objects\":["
    for key in $keys; do
        delete_items="$delete_items{\"Key\": \"$key\"},\"
    done
    delete_items=${delete_items%?} # Remove the final comma.
    delete_items="$delete_items]"

    response=$(aws s3api delete-objects \
        --bucket "$bucket_name" \
        --delete "$delete_items")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-object operation failed.\n
$response"
        return 1
    fi
}
```

- Per i dettagli sull'API, consulta [DeleteObjects AWS CLI](#) Command Reference.

C++

SDK per C++

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
bool AwsDoc::S3::deleteObjects(const std::vector<Aws::String> &objectKeys,
                                const Aws::String &fromBucket,
                                const Aws::S3::S3ClientConfiguration
                                &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectsRequest request;

    Aws::S3::Model::Delete deleteObject;
    for (const Aws::String &objectKey: objectKeys) {

deleteObject.AddObjects(Aws::S3::Model::ObjectIdentifier().WithKey(objectKey));
    }

    request.SetDelete(deleteObject);
    request.SetBucket(fromBucket);

    Aws::S3::Model::DeleteObjectsOutcome outcome =
        client.DeleteObjects(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error deleting objects. " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Successfully deleted the objects.";
        for (size_t i = 0; i < objectKeys.size(); ++i) {
            std::cout << objectKeys[i];
            if (i < objectKeys.size() - 1) {
                std::cout << ", ";
            }
        }
    }
}
```



```
        std::cout << " from bucket " << fromBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Per i dettagli sull'API, [DeleteObjects](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Il comando seguente elimina un oggetto da un bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api delete-objects --bucket amzn-s3-demo-bucket --delete file://delete.json
```

`delete.json` è un documento JSON nella directory corrente che specifica l'oggetto da eliminare.

```
{
  "Objects": [
    {
      "Key": "test1.txt"
    }
  ],
  "Quiet": false
}
```

Output:

```
{
  "Deleted": [
    {
      "DeleteMarkerVersionId": "mYAT5Mc6F7aeUL8SS7FAAqUP01koHwzU",
      "Key": "test1.txt",
      "DeleteMarker": true
    }
  ]
}
```

- Per i dettagli sull'API, consulta [DeleteObjects AWS CLI Command Reference](#).

Go

SDK per Go V2

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// DeleteObjects deletes a list of objects from a bucket.  
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects  
    []types.ObjectIdentifier, bypassGovernance bool) error {  
    if len(objects) == 0 {  
        return nil  
    }  
}
```

```
input := s3.DeleteObjectsInput{
    Bucket: aws.String(bucket),
    Delete: &types.Delete{
        Objects: objects,
        Quiet:    aws.Bool(true),
    },
}
if bypassGovernance {
    input.BypassGovernanceRetention = aws.Bool(true)
}
delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
if err != nil || len(delOut.Errors) > 0 {
    log.Printf("Error deleting objects from bucket %s.\n", bucket)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    } else if len(delOut.Errors) > 0 {
        for _, outErr := range delOut.Errors {
            log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
        }
        err = fmt.Errorf("%s", *delOut.Errors[0].Message)
    }
} else {
    for _, delObjs := range delOut.Deleted {
        err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObjs.Key},
            time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to be deleted.\n",
                *delObjs.Key)
        } else {
            log.Printf("Deleted %s.\n", *delObjs.Key)
        }
    }
}
return err
}
```

- Per i dettagli sull'API, [DeleteObjects](#) consulta AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */

public class DeleteMultiObjects {
    public static void main(String[] args) {
        final String usage = ""

        Usage:    <bucketName>

        Where:
            bucketName - the Amazon S3 bucket name.
```

```
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    deleteBucketObjects(s3, bucketName);
    s3.close();
}

/**
 * Deletes multiple objects from an Amazon S3 bucket.
 *
 * @param s3 An Amazon S3 client object.
 * @param bucketName The name of the Amazon S3 bucket to delete objects from.
 */
public static void deleteBucketObjects(S3Client s3, String bucketName) {
    // Upload three sample objects to the specified Amazon S3 bucket.
    ArrayList<ObjectIdentifier> keys = new ArrayList<>();
    PutObjectRequest putOb;
    ObjectIdentifier objectId;

    for (int i = 0; i < 3; i++) {
        String keyName = "delete object example " + i;
        objectId = ObjectIdentifier.builder()
            .key(keyName)
            .build();

        putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        s3.putObject(putOb, RequestBody.fromString(keyName));
        keys.add(objectId);
    }
}
```

```
        System.out.println(keys.size() + " objects successfully created.");

        // Delete multiple objects in one request.
        Delete del = Delete.builder()
            .objects(keys)
            .build();

        try {
            DeleteObjectsRequest multiObjectDeleteRequest =
                DeleteObjectsRequest.builder()
                    .bucket(bucketName)
                    .delete(del)
                    .build();

            s3.deleteObjects(multiObjectDeleteRequest);
            System.out.println("Multiple objects are deleted!");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [DeleteObjects](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina più oggetti.

```
import {
    DeleteObjectsCommand,
    S3Client,
```

```

    S3ServiceException,
    waitUntilObjectNotExists,
  } from "@aws-sdk/client-s3";

/**
 * Delete multiple objects from an S3 bucket.
 * @param {{ bucketName: string, keys: string[] }}
 */
export const main = async ({ bucketName, keys }) => {
  const client = new S3Client({});

  try {
    const { Deleted } = await client.send(
      new DeleteObjectsCommand({
        Bucket: bucketName,
        Delete: {
          Objects: keys.map((k) => ({ Key: k })),
        },
      }),
    );
    for (const key in keys) {
      await waitUntilObjectNotExists(
        { client },
        { Bucket: bucketName, Key: key },
      );
    }
    console.log(
      `Successfully deleted ${Deleted.length} objects from S3 bucket. Deleted
objects:`,
    );
    console.log(Deleted.map((d) => ` • ${d.Key}`).join("\n"));
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting objects from ${bucketName}. The bucket
doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting objects from ${bucketName}.
${caught.name}: ${caught.message}`,
      );
    }
  }
};

```

```
    );  
  } else {  
    throw caught;  
  }  
}  
};
```

- Per i dettagli sull'API, [DeleteObjects](#) consulta AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun deleteBucketObjects(  
    bucketName: String,  
    objectName: String,  
) {  
    val objectId =  
        ObjectIdentifier {  
            key = objectName  
        }  
  
    val delOb =  
        Delete {  
            objects = listOf(objectId)  
        }  
  
    val request =  
        DeleteObjectsRequest {  
            bucket = bucketName  
            delete = delOb  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.deleteObjects(request)
```



```
        println("$objectName was deleted from $bucketName")
    }
}
```

- Per i dettagli sull'API, [DeleteObjects](#) consulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina un set di oggetti da un elenco di chiavi.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $objects = [];
    foreach ($contents['Contents'] as $content) {
        $objects[] = [
            'Key' => $content['Key'],
        ];
    }
    $this->s3client->deleteObjects([
        'Bucket' => $this->bucketName,
        'Delete' => [
            'Objects' => $objects,
        ],
    ]);
    $check = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    if (isset($check['Contents']) && count($check['Contents']) > 0) {
        throw new Exception("Bucket wasn't empty.");
    }
    echo "Deleted all objects and folders from $this->bucketName.\n";
} catch (Exception $exception) {
```

```
        echo "Failed to delete $fileName from $this->bucketName with error:
" . $exception->getMessage();
        exit("Please fix error with object deletion before continuing.");
    }
}
```

- Per i dettagli sull'API, [DeleteObjects](#) consulta AWS SDK per PHP API Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando rimuove l'oggetto "sample.txt" dal bucket "test-files". Prima di eseguire il comando, viene richiesto di confermare l'operazione. Per disabilitare la richiesta, utilizza il parametro -Force.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Esempio 2: questo comando rimuove la versione specificata dell'oggetto "sample.txt" dal bucket "test-files", supponendo che il bucket sia stato configurato per abilitare le versioni degli oggetti.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Esempio 3: questo comando rimuove gli oggetti "sample1.txt", "sample2.txt" e "sample3.txt" dal bucket "test-files" in un'unica operazione batch. La risposta del servizio elencherà tutte le chiavi elaborate, indipendentemente dallo stato di riuscita o errore dell'eliminazione. Per ottenere solo gli errori relativi alle chiavi che non hanno potuto essere elaborate dal servizio, aggiungi il ReportErrorsOnly parametro - (questo parametro può essere specificato anche con l'alias -Quiet).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
"sample2.txt", "sample3.txt" )
```

Esempio 4: Questo esempio utilizza un'espressione in linea con il KeyCollection parametro - per ottenere le chiavi degli oggetti da eliminare. Get-S3Object restituisce una raccolta di Amazon.S3.Model.S3Object istanze, ognuna delle quali ha un membro Key di tipo string che identifica l'oggetto.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object  
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Esempio 5: Questo esempio ottiene tutti gli oggetti che hanno un prefisso chiave prefix/subprefix "" nel bucket e li elimina. Nota che gli oggetti in entrata vengono elaborati uno alla volta. Per raccolte di grandi dimensioni, è consigliabile passare la raccolta al parametro -InputObject (alias -S3ObjectCollection) del cmdlet per consentire l'eliminazione in batch con una singola chiamata al servizio.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |  
Remove-S3Object -Force
```

Esempio 6: Questo esempio reindirizza una raccolta di Amazon.S3.Model.S3ObjectVersion istanze che rappresentano marcatori di eliminazione al cmdlet per l'eliminazione. Nota che gli oggetti in entrata vengono elaborati uno alla volta. Per raccolte di grandi dimensioni, è consigliabile passare la raccolta al parametro -InputObject (alias -S3ObjectCollection) del cmdlet per consentire l'eliminazione in batch con una singola chiamata al servizio.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where  
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Esempio 7: questo script mostra come eseguire un'eliminazione in batch di un set di oggetti (in questo caso eliminare i marker) creando una matrice di oggetti da utilizzare con il parametro -.KeyAndVersionCollection

```
$keyVersions = @()  
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where  
{$_ .IsDeleteMarker -eq "True"}  
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =  
$marker.VersionId } }  
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -  
Force
```

- Per i dettagli sull'API, vedere [DeleteObjects](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando rimuove l'oggetto "sample.txt" dal bucket "test-files". Prima di eseguire il comando, viene richiesto di confermare l'operazione. Per disabilitare la richiesta, utilizza il parametro -Force.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Esempio 2: questo comando rimuove la versione specificata dell'oggetto "sample.txt" dal bucket "test-files", supponendo che il bucket sia stato configurato per abilitare le versioni degli oggetti.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId  
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Esempio 3: questo comando rimuove gli oggetti "sample1.txt", "sample2.txt" e "sample3.txt" dal bucket "test-files" in un'unica operazione batch. La risposta del servizio elencherà tutte le chiavi elaborate, indipendentemente dallo stato di riuscita o errore dell'eliminazione. Per ottenere solo gli errori relativi alle chiavi che non hanno potuto essere elaborate dal servizio, aggiungi il ReportErrorsOnly parametro - (questo parametro può essere specificato anche con l'alias -Quiet).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",  
"sample2.txt", "sample3.txt" )
```

Esempio 4: Questo esempio utilizza un'espressione in linea con il KeyCollection parametro - per ottenere le chiavi degli oggetti da eliminare. Get-S3Object restituisce una raccolta di Amazon.S3.Model.S3Object istanze, ognuna delle quali ha un membro Key di tipo string che identifica l'oggetto.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object  
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Esempio 5: Questo esempio ottiene tutti gli oggetti che hanno un prefisso chiave prefix/subprefix "" nel bucket e li elimina. Nota che gli oggetti in entrata vengono elaborati uno alla volta. Per raccolte di grandi dimensioni, è consigliabile passare la raccolta al parametro -InputObject (alias -S3ObjectCollection) del cmdlet per consentire l'eliminazione in batch con una singola chiamata al servizio.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |  
Remove-S3Object -Force
```

Esempio 6: Questo esempio reindirizza una raccolta di `Amazon.S3.Model.S3ObjectVersion` istanze che rappresentano marcatori di eliminazione al cmdlet per l'eliminazione. Nota che gli oggetti in entrata vengono elaborati uno alla volta. Per raccolte di grandi dimensioni, è consigliabile passare la raccolta al parametro `-InputObject` (alias `-S3ObjectCollection`) del cmdlet per consentire l'eliminazione in batch con una singola chiamata al servizio.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where  
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Esempio 7: questo script mostra come eseguire un'eliminazione in batch di un set di oggetti (in questo caso eliminare i marker) creando una matrice di oggetti da utilizzare con il parametro `-KeyAndVersionCollection`

```
$keyVersions = @()  
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where  
{$_ .IsDeleteMarker -eq "True"}  
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =  
$marker.VersionId } }  
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -  
Force
```

- Per i dettagli sull'API, vedere [DeleteObjects](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Python

SDK per Python (Boto3)

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina un set di oggetti utilizzando un elenco di chiavi oggetto.

```

class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                                that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    @staticmethod
    def delete_objects(bucket, object_keys):
        """
        Removes a list of objects from a bucket.
        This operation is done as a batch in a single request.

        :param bucket: The bucket that contains the objects. This is a Boto3
Bucket
                                resource.
        :param object_keys: The list of keys that identify the objects to remove.
        :return: The response that contains data about which objects were deleted
and any that could not be deleted.
        """
        try:
            response = bucket.delete_objects(
                Delete={"Objects": [{"Key": key} for key in object_keys]}
            )
            if "Deleted" in response:
                logger.info(
                    "Deleted objects '%s' from bucket '%s'.",
                    [del_obj["Key"] for del_obj in response["Deleted"]],
                    bucket.name,
                )
            if "Errors" in response:
                logger.warning(
                    "Could not delete objects '%s' from bucket '%s'.",
                    [
                        f"{del_obj['Key']}: {del_obj['Code']}"
                        for del_obj in response["Errors"]
                    ],
                )

```

```

        bucket.name,
    )
except ClientError:
    logger.exception("Couldn't delete any objects from bucket %s.",
bucket.name)
    raise
else:
    return response

```

Elimina tutti gli oggetti in un bucket.

```

class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                                that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    @staticmethod
    def empty_bucket(bucket):
        """
        Remove all objects from a bucket.

        :param bucket: The bucket to empty. This is a Boto3 Bucket resource.
        """
        try:
            bucket.objects.delete()
            logger.info("Emptied bucket '%s'.", bucket.name)
        except ClientError:
            logger.exception("Couldn't empty bucket '%s'.", bucket.name)
            raise

```

Elimina in modo permanente un oggetto con versione eliminando tutte le relative versioni.

```
def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

- Per i dettagli sull'API, consulta [DeleteObjects AWS SDK for Python \(Boto3\) API Reference](#).

Ruby

SDK per Ruby

Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?")
  answer = gets.chomp.downcase
  if answer == 'y'
    bucket.objects.batch_delete!
```



```

    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

```

- Per i dettagli sull'API, [DeleteObjects](#) consulta AWS SDK per Ruby API Reference.

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

/// Delete the objects in a bucket.
pub async fn delete_objects(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    objects_to_delete: Vec<String>,
) -> Result<(), S3ExampleError> {
    // Push into a mut vector to use '?' early return errors while building
    object keys.
    let mut delete_object_ids: Vec<aws_sdk_s3::types::ObjectIdentifier> = vec![];
    for obj in objects_to_delete {
        let obj_id = aws_sdk_s3::types::ObjectIdentifier::builder()
            .key(obj)
            .build()
            .map_err(|err| {
                S3ExampleError::new(format!("Failed to build key for
delete_object: {err:?}"))
            })?;
        delete_object_ids.push(obj_id);
    }
}

```

```

    client
      .delete_objects()
      .bucket(bucket_name)
      .delete(
        aws_sdk_s3::types::Delete::builder()
          .set_objects(Some(delete_object_ids))
          .build()
          .map_err(|err| {
            S3ExampleError::new(format!("Failed to build delete_object
input {err:?}"))
          })?,
      )
      .send()
      .await?;
    Ok(())
  }
}

```

- Per i dettagli sulle API, consulta la [DeleteObjects](#) guida di riferimento all'API AWS SDK for Rust.

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

TRY.
  oo_result = lo_s3->deleteobjects(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name
    io_delete = NEW /aws1/cl_s3_delete( it_objects = it_object_keys ) ).
  MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- Per i dettagli sulle API, [DeleteObjects](#) consulta AWS SDK for SAP ABAP API reference.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import AWSS3

public func deleteObjects(bucket: String, keys: [String]) async throws {
    let input = DeleteObjectsInput(
        bucket: bucket,
        delete: S3ClientTypes.Delete(
            objects: keys.map { S3ClientTypes.ObjectIdentifier(key: $0) },
            quiet: true
        )
    )

    do {
        _ = try await client.deleteObjects(input: input)
    } catch {
        print("ERROR: deleteObjects:", dump(error))
        throw error
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteObjects](#) guida di riferimento all'API AWS SDK for Swift.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeletePublicAccessBlock** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare DeletePublicAccessBlock.

CLI

AWS CLI

Come eliminare la configurazione di blocco dell'accesso pubblico per un bucket

L'esempio delete-public-access-block seguente rimuove la configurazione del blocco dell'accesso pubblico per il bucket e l'ID specificati.

```
aws s3api delete-public-access-block \  
  --bucket amzn-s3-demo-bucket
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [DeletePublicAccessBlock AWS CLI Command Reference](#).

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando disattiva l'impostazione di blocco dell'accesso pubblico per il bucket specificato.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select  
'^BucketName'
```

Output:

```
amzn-s3-demo-bucket
```

- Per i dettagli sull'API, vedere [DeletePublicAccessBlock](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando disattiva l'impostazione di blocco dell'accesso pubblico per il bucket specificato.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select  
'^BucketName'
```

Output:

```
amzn-s3-demo-bucket
```

- Per i dettagli sull'API, vedere [DeletePublicAccessBlock](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketAccelerateConfiguration** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare GetBucketAccelerateConfiguration.

CLI

AWS CLI

Come recuperare la configurazione di accelerazione di un bucket

L'esempio get-bucket-accelerate-configuration seguente recupera la configurazione di accelerazione per il bucket specificato.

```
aws s3api get-bucket-accelerate-configuration \  
--bucket amzn-s3-demo-bucket
```

Output:

```
{  
  "Status": "Enabled"  
}
```

- Per i dettagli sull'API, consulta [GetBucketAccelerateConfiguration AWS CLI Command Reference](#).

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando restituisce il valore Enabled se le impostazioni di accelerazione del trasferimento sono abilitate per il bucket specificato.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Value
-----
Enabled
```

- Per i dettagli sull'API, vedere [GetBucketAccelerateConfiguration](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando restituisce il valore Enabled se le impostazioni di accelerazione del trasferimento sono abilitate per il bucket specificato.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Value
-----
Enabled
```

- Per i dettagli sull'API, vedere [GetBucketAccelerateConfiguration](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketAc1** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare GetBucketAc1.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- [Controllare se un bucket esiste](#)
- [Gestire le liste di controllo degli accessi \(ACL\)](#)

.NET

SDK per .NET

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>
/// Get the access control list (ACL) for the new bucket.
/// </summary>
/// <param name="client">The initialized client object used to get the
/// access control list (ACL) of the bucket.</param>
/// <param name="newBucketName">The name of the newly created bucket.</
param>
/// <returns>An S3AccessControlList.</returns>
public static async Task<S3AccessControlList>
GetACLForBucketAsync(IAmazonS3 client, string newBucketName)
{
    // Retrieve bucket ACL to show that the ACL was properly applied to
    // the new bucket.
    GetACLResponse getACLResponse = await client.GetACLAsync(new
GetACLRequest
    {
        BucketName = newBucketName,
    });

    return getACLResponse.AccessControlList;
}
```

- Per i dettagli sull'API, [GetBucketAcl](#) consulta AWS SDK per .NET API Reference.

C++

SDK per C++

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
bool AwsDoc::S3::getBucketAcl(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
        s3Client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketAcl: "
                    << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
            outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            std::cout << "For bucket " << bucketName << ": "
                      << std::endl << std::endl;

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
```



```

        << getGranteeTypeString(grantee.GetType()) <<
std::endl;
    }

    if (grantee.DisplayNameHasBeenSet()) {
        std::cout << "Display name: "
        << grantee.GetDisplayName() << std::endl;
    }

    if (grantee.EmailAddressHasBeenSet()) {
        std::cout << "Email address: "
        << grantee.GetEmailAddress() << std::endl;
    }

    if (grantee.IDHasBeenSet()) {
        std::cout << "ID: "
        << grantee.GetID() << std::endl;
    }

    if (grantee.URIHasBeenSet()) {
        std::cout << "URI: "
        << grantee.GetURI() << std::endl;
    }

    std::cout << "Permission: " <<
        getPermissionString(grant.GetPermission()) <<
        std::endl << std::endl;
    }
}

return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string.
 */

Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
    }
}

```

```

        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
    string.
/*!
    \param permission: Permission enumeration.
    \return String: Human-readable string.
*/

Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can list objects in this bucket, create/overwrite/delete "
                "objects in this bucket, and read/write this "
                "bucket's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can list objects in this bucket";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this bucket's permissions";
        case Aws::S3::Model::Permission::WRITE:
            return "Can create, overwrite, and delete objects in this bucket";
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this bucket's permissions";
        default:
            return "Permission unknown";
    }

    return "Permission unknown";
}

```

- Per i dettagli sull'API, [GetBucketAcl](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Il comando seguente recupera una lista di controllo degli accessi (ACL) per un bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "Owner": {
    "DisplayName": "my-username",
    "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
  },
  "Grants": [
    {
      "Grantee": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
      },
      "Permission": "FULL_CONTROL"
    }
  ]
}
```

- Per i dettagli sull'API, consulta [GetBucketAcl AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetAcl {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <bucketName> <objectKey>

            Where:
            bucketName - The Amazon S3 bucket to get the access control list
(ACL) for.
            objectKey - The object to get the ACL for.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        System.out.println("Retrieving ACL for object: " + objectKey);
        System.out.println("in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
```

```

        .build();

        getBucketACL(s3, objectKey, bucketName);
        s3.close();
        System.out.println("Done!");
    }

    /**
     * Retrieves the Access Control List (ACL) for an object in an Amazon S3
     bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param objectKey The key of the object for which the ACL is to be
     retrieved.
     * @param bucketName The name of the bucket containing the object.
     * @return The ID of the grantee who has permission on the object, or an
     empty string if an error occurs.
     */
    public static String getBucketACL(S3Client s3, String objectKey, String
    bucketName) {
        try {
            GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
            List<Grant> grants = aclRes.grants();
            String grantee = "";
            for (Grant grant : grants) {
                System.out.format("  %s: %s\n", grant.grantee().id(),
    grant.permission());
                grantee = grant.grantee().id();
            }

            return grantee;
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}

```

- Per i dettagli sull'API, [GetBucketAcl](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Recupera le autorizzazioni ACL.

```
import {
  GetBucketAclCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Retrieves the Access Control List (ACL) for an S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetBucketAclCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`ACL for bucket "${bucketName}":`);
    console.log(JSON.stringify(response, null, 2));
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
```

```
        console.error(
            `Error from S3 while getting ACL for ${bucketName}. The bucket doesn't
            exist.`
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while getting ACL for ${bucketName}. ${caught.name}:
            ${caught.message}`
        );
    } else {
        throw caught;
    }
}
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [GetBucketAcl](#) consulta AWS SDK per JavaScript API Reference.

PowerShell

Strumenti per PowerShell V5

Esempio 1: questo comando ottiene i dettagli del proprietario dell'oggetto S3.

```
(Get-S3BucketACL -BucketName 'amzn-s3-demo-bucket' -Select *).Owner
```

Output:

```
DisplayName Id
----- --
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- Per i dettagli sull'API, vedere [GetBucketAcl](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Python

SDK per Python (Boto3)

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def get_acl(self):
        """
        Get the ACL of the bucket.

        :return: The ACL of the bucket.
        """
        try:
            acl = self.bucket.Acl()
            logger.info(
                "Got ACL for bucket %s. Owner is %s.", self.bucket.name,
                acl.owner
            )
        except ClientError:
            logger.exception("Couldn't get ACL for bucket %s.", self.bucket.name)
            raise
        else:
            return acl
```


- Per i dettagli sull'API, consulta [GetBucketAcl AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.  
    oo_result = lo_s3->getbucketacl(          " oo_result is returned for  
testing purposes. "  
    iv_bucket = iv_bucket_name ).  
    MESSAGE 'Retrieved bucket ACL.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Per i dettagli sulle API, [GetBucketAcl](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketAnalyticsConfiguration** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare GetBucketAnalyticsConfiguration.

CLI

AWS CLI

Come recuperare la configurazione di analisi per un bucket con un ID specifico

L'esempio get-bucket-analytics-configuration seguente visualizza la configurazione di analisi per il bucket e l'ID specificati.

```
aws s3api get-bucket-analytics-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --id 1
```

Output:

```
{  
  "AnalyticsConfiguration": {  
    "StorageClassAnalysis": {},  
    "Id": "1"  
  }  
}
```

- Per i dettagli sull'API, consulta [GetBucketAnalyticsConfiguration AWS CLI Command Reference](#).

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando restituisce i dettagli del filtro di analisi denominato “testfilter” nel bucket S3 specificato.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId  
'testfilter'
```

- Per i dettagli sull'API, vedere [GetBucketAnalyticsConfiguration](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando restituisce i dettagli del filtro di analisi denominato “testfilter” nel bucket S3 specificato.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId  
'testfilter'
```

- Per i dettagli sull'API, vedere [GetBucketAnalyticsConfiguration](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketCors** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare GetBucketCors.

.NET

SDK per .NET

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>
/// Retrieve the CORS configuration applied to the Amazon S3 bucket.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used
/// to retrieve the CORS configuration.</param>
/// <returns>The created CORS configuration object.</returns>
private static async Task<CORSConfiguration>
RetrieveCORSConfigurationAsync(AmazonS3Client client)
{
    GetCORSConfigurationRequest request = new
GetCORSConfigurationRequest()
    {
        BucketName = BucketName,
    };
    var response = await client.GetCORSConfigurationAsync(request);
    var configuration = response.Configuration;
    PrintCORSRules(configuration);
    return configuration;
}
```

- Per i dettagli sull'API, [GetBucketCors](#) consulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Il comando seguente recupera la configurazione di condivisione Cross-Origin delle risorse per un bucket denominato: `amzn-s3-demo-bucket`

```
aws s3api get-bucket-cors --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "CORSRules": [
    {
      "AllowedHeaders": [
        "*"
      ],
      "ExposeHeaders": [
        "x-amz-server-side-encryption"
      ],
      "AllowedMethods": [
        "PUT",
        "POST",
        "DELETE"
      ],
      "MaxAgeSeconds": 3000,
      "AllowedOrigins": [
        "http://www.example.com"
      ]
    },
    {
      "AllowedHeaders": [
        "Authorization"
      ],
      "MaxAgeSeconds": 3000,
      "AllowedMethods": [
        "GET"
      ],
      "AllowedOrigins": [
        "*"
      ]
    }
  ]
}
```

```
}
```

- Per i dettagli sull'API, vedere [GetBucketCors](#) in AWS CLI Command Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Recupera la policy CORS per il bucket.

```
import {
  GetBucketCorsCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Log the Cross-Origin Resource Sharing (CORS) configuration information
 * set for the bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new GetBucketCorsCommand({
    Bucket: bucketName,
  });

  try {
    const { CORSRules } = await client.send(command);
    console.log(JSON.stringify(CORSRules));
    CORSRules.forEach((cr, i) => {
      console.log(
        `\nCORSRule ${i + 1}`,
        `\n${"-"}.repeat(10)`,
        `\nAllowedHeaders: ${cr.AllowedHeaders}`,
        `\nAllowedMethods: ${cr.AllowedMethods}`,
      );
    });
  } catch (err) {
    if (err instanceof S3ServiceException) {
      console.log(`Error: ${err.message}`);
    } else {
      console.log(`Error: ${err}`);
    }
  }
};
```

```

        '\nAllowedOrigins: ${cr.AllowedOrigins}`',
        '\nExposeHeaders: ${cr.ExposeHeaders}`',
        '\nMaxAgeSeconds: ${cr.MaxAgeSeconds}`',
    );
});
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while getting bucket CORS rules for ${bucketName}. The
            bucket doesn't exist.`
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while getting bucket CORS rules for ${bucketName}.
            ${caught.name}: ${caught.message}`
        );
    } else {
        throw caught;
    }
}
};

```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [GetBucketCors](#) consulta AWS SDK per JavaScript API Reference.

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

```

```
def __init__(self, bucket):
    """
    :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                    that wraps bucket actions in a class-like structure.
    """
    self.bucket = bucket
    self.name = bucket.name

def get_cors(self):
    """
    Get the CORS rules for the bucket.

    :return The CORS rules for the specified bucket.
    """
    try:
        cors = self.bucket.Cors()
        logger.info(
            "Got CORS rules %s for bucket '%s'.", cors.cors_rules,
self.bucket.name
        )
    except ClientError:
        logger.exception(("Couldn't get CORS for bucket %s.",
self.bucket.name))
        raise
    else:
        return cors
```

- Per i dettagli sull'API, consulta [GetBucketCors AWS SDK for Python \(Boto3\) API Reference](#).

Ruby

SDK per Ruby

Note

C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS
  configuration for the bucket.
  def cors
    @bucket_cors.data
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
  why: #{e.message}"
    nil
  end
end
```

- Per i dettagli sull'API, [GetBucketCors](#) consulta AWS SDK per Ruby API Reference.

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

TRY.


```

        oo_result = lo_s3->getbucketcors(           " oo_result is returned for
testing purposes. "
        iv_bucket = iv_bucket_name ).
        MESSAGE 'Retrieved bucket CORS configuration.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    ENDTRY.

```

- Per i dettagli sulle API, [GetBucketCors](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketEncryption** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare GetBucketEncryption.

.NET

SDK per .NET

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

/// <summary>
/// Get and print the encryption settings of a bucket.
/// </summary>
/// <param name="bucketName">Name of the bucket.</param>
/// <returns>Async task.</returns>
public static async Task GetEncryptionSettings(string bucketName)
{
    // Check and print the bucket encryption settings.
    Console.WriteLine($"Getting encryption settings for bucket
{bucketName}.");

    try

```

```

    {
        var settings =
            await _s3Client.GetBucketEncryptionAsync(
                new GetBucketEncryptionRequest() { BucketName =
bucketName });

        foreach (var encryptionSettings in
settings?.ServerSideEncryptionConfiguration?.ServerSideEncryptionRules!)
        {
            Console.WriteLine(
                $"{Algorithm:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionAlgorithm}");
            Console.WriteLine(
                $"{Key:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionKeyManagementService
}
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine(ex.ErrorCode == "InvalidBucketName"
                ? $"Bucket {bucketName} was not found."
                : $"Unable to get bucket encryption for bucket {bucketName},
{ex.Message}");
        }
    }
}

```

- Per i dettagli sull'API, [GetBucketEncryption](#) consulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Come recuperare la configurazione di crittografia lato server di un bucket

L'esempio `get-bucket-encryption` seguente restituisce la configurazione di crittografia lato server per il bucket `amzn-s3-demo-bucket`.

```
aws s3api get-bucket-encryption \
    --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "ServerSideEncryptionConfiguration": {
    "Rules": [
      {
        "ApplyServerSideEncryptionByDefault": {
          "SSEAlgorithm": "AES256"
        }
      }
    ]
  }
}
```

- Per i dettagli sull'API, consulta [GetBucketEncryption AWS CLI Command Reference](#).

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando restituisce tutte le regole di crittografia lato server associate al bucket specificato.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [GetBucketEncryption](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando restituisce tutte le regole di crittografia lato server associate al bucket specificato.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [GetBucketEncryption](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketInventoryConfiguration** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare `GetBucketInventoryConfiguration`.

CLI

AWS CLI

Come recuperare la configurazione di inventario per un bucket

L'esempio `get-bucket-inventory-configuration` seguente recupera la configurazione dell'inventario con ID 1 per il bucket specificato.

```
aws s3api get-bucket-inventory-configuration \
  --bucket amzn-s3-demo-bucket \
  --id 1
```

Output:

```
{
  "InventoryConfiguration": {
    "IsEnabled": true,
    "Destination": {
      "S3BucketDestination": {
        "Format": "ORC",
        "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
        "AccountId": "123456789012"
      }
    },
    "IncludedObjectVersions": "Current",
    "Id": "1",
    "Schedule": {
      "Frequency": "Weekly"
    }
  }
}
```

- Per i dettagli sull'API, consulta [GetBucketInventoryConfiguration AWS CLI Command Reference](#).

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando restituisce i dettagli dell'inventario denominato "testinventory" per il bucket S3 specificato.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId 'testinventory'
```

- Per i dettagli sull'API, vedere [GetBucketInventoryConfiguration](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando restituisce i dettagli dell'inventario denominato "testinventory" per il bucket S3 specificato.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId 'testinventory'
```

- Per i dettagli sull'API, vedere [GetBucketInventoryConfiguration](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketLifecycleConfiguration** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare GetBucketLifecycleConfiguration.

.NET

SDK per .NET

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

    /// <summary>
    /// Returns a configuration object for the supplied bucket name.
    /// </summary>
    /// <param name="client">The S3 client object used to call
    /// the GetLifecycleConfigurationAsync method.</param>
    /// <param name="bucketName">The name of the S3 bucket for which a
    /// configuration will be created.</param>
    /// <returns>Returns a new LifecycleConfiguration object.</returns>
    public static async Task<LifecycleConfiguration>
    RetrieveLifecycleConfigAsync(IAmazonS3 client, string bucketName)
    {
        var request = new GetLifecycleConfigurationRequest()
        {
            BucketName = bucketName,
        };
        var response = await client.GetLifecycleConfigurationAsync(request);
        var configuration = response.Configuration;
        return configuration;
    }

```

- Per i dettagli sull'API, [GetBucketLifecycleConfiguration](#) consulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Il comando seguente recupera una configurazione del ciclo di vita per un bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket
```

Output:

```

{
  "Rules": [
    {
      "ID": "Move rotated logs to Glacier",

```

```

        "Prefix": "rotated/",
        "Status": "Enabled",
        "Transitions": [
            {
                "Date": "2015-11-10T00:00:00.000Z",
                "StorageClass": "GLACIER"
            }
        ],
    },
    {
        "Status": "Enabled",
        "Prefix": "",
        "NoncurrentVersionTransitions": [
            {
                "NoncurrentDays": 0,
                "StorageClass": "GLACIER"
            }
        ],
        "ID": "Move old versions to Glacier"
    }
]
}

```

- Per i dettagli sull'API, consulta [GetBucketLifecycleConfiguration AWS CLI Command Reference](#).

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """

```

```
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def get_lifecycle_configuration(self):
        """
        Get the lifecycle configuration of the bucket.

        :return: The lifecycle rules of the specified bucket.
        """
        try:
            config = self.bucket.LifecycleConfiguration()
            logger.info(
                "Got lifecycle rules %s for bucket '%s'.",
                config.rules,
                self.bucket.name,
            )
        except:
            logger.exception(
                "Couldn't get lifecycle rules for bucket '%s'.", self.bucket.name
            )
            raise
        else:
            return config.rules
```

- Per i dettagli sull'API, consulta [GetBucketLifecycleConfiguration AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.  
    oo_result = lo_s3->getbucketlifecycleconf(      " oo_result is  
returned for testing purposes. "  
    iv_bucket = iv_bucket_name ).  
    MESSAGE 'Retrieved bucket lifecycle configuration.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Per i dettagli sulle API, [GetBucketLifecycleConfiguration](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketLocation** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare GetBucketLocation.

CLI

AWS CLI

Il comando seguente recupera il vincolo di posizione per un bucket denominato `amzn-s3-demo-bucket`, se esiste un vincolo.

```
aws s3api get-bucket-location --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "LocationConstraint": "us-west-2"
}
```

- Per i dettagli sull'API, consulta [GetBucketLocation AWS CLI](#) Command Reference.

PowerShell**Strumenti per PowerShell V4**

Esempio 1: questo comando restituisce il vincolo di posizione eventualmente presente per il bucket "amzn-s3-demo-bucket".

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Value
-----
ap-south-1
```

- Per i dettagli sull'API, vedere [GetBucketLocation](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando restituisce il vincolo di posizione eventualmente presente per il bucket "amzn-s3-demo-bucket".

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Value
-----
ap-south-1
```

- Per i dettagli sull'API, vedere [GetBucketLocation](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
async fn show_buckets(
    strict: bool,
    client: &Client,
    region: BucketLocationConstraint,
) -> Result<(), S3ExampleError> {
    let mut buckets = client.list_buckets().into_paginator().send();

    let mut num_buckets = 0;
    let mut in_region = 0;

    while let Some(Ok(output)) = buckets.next().await {
        for bucket in output.buckets() {
            num_buckets += 1;
            if strict {
                let r = client
                    .get_bucket_location()
                    .bucket(bucket.name().unwrap_or_default())
                    .send()
                    .await?;

                if r.location_constraint() == Some(&region) {
                    println!("{}", bucket.name().unwrap_or_default());
                    in_region += 1;
                }
            } else {
                println!("{}", bucket.name().unwrap_or_default());
            }
        }
    }

    println!();
    if strict {
```

```
println!(
    "Found {} buckets in the {} region out of a total of {} buckets.",
    in_region, region, num_buckets
);
} else {
    println!("Found {} buckets in all regions.", num_buckets);
}

Ok(())
}
```

- Per i dettagli sulle API, consulta la [GetBucketLocation](#) guida di riferimento all'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketLogging** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare GetBucketLogging.

CLI

AWS CLI

Come recuperare lo stato della registrazione di log di un bucket

L'esempio get-bucket-logging seguente recupera lo stato di registrazione per il bucket specificato.

```
aws s3api get-bucket-logging \
    --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "LoggingEnabled": {
    "TargetPrefix": "",
    "TargetBucket": "amzn-s3-demo-bucket-logs"
```

```
}
}
```

- Per i dettagli sull'API, consulta [GetBucketLogging AWS CLI](#) Command Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando restituisce lo stato di registrazione per il bucket specificato.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Output:

TargetBucketName	Grants	TargetPrefix
-----	-----	-----
testbucket1	{}	testprefix

- Per i dettagli sull'API, vedere [GetBucketLogging](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando restituisce lo stato di registrazione per il bucket specificato.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Output:

TargetBucketName	Grants	TargetPrefix
-----	-----	-----
testbucket1	{}	testprefix

- Per i dettagli sull'API, vedere [GetBucketLogging](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketMetricsConfiguration** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare `GetBucketMetricsConfiguration`.

CLI

AWS CLI

Come recuperare la configurazione delle metriche per un bucket con un ID specifico

L'esempio `get-bucket-metrics-configuration` seguente visualizza la configurazione delle metriche per il bucket e l'ID specificati.

```
aws s3api get-bucket-metrics-configuration \
  --bucket amzn-s3-demo-bucket \
  --id 123
```

Output:

```
{
  "MetricsConfiguration": {
    "Filter": {
      "Prefix": "logs"
    },
    "Id": "123"
  }
}
```

- Per i dettagli sull'API, consulta [GetBucketMetricsConfiguration AWS CLI Command Reference](#).

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando restituisce i dettagli sul filtro delle metriche denominato "testfilter" per il bucket S3 specificato.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
'testfilter'
```

- Per i dettagli sull'API, vedere [GetBucketMetricsConfiguration](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando restituisce i dettagli sul filtro delle metriche denominato “testfilter” per il bucket S3 specificato.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId 'testfilter'
```

- Per i dettagli sull'API, vedere [GetBucketMetricsConfiguration](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketNotification** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare GetBucketNotification.

CLI

AWS CLI

Il comando seguente recupera la configurazione di notifica per un bucket denominato amzn-s3-demo-bucket.

```
aws s3api get-bucket-notification --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "TopicConfiguration": {
    "Topic": "arn:aws:sns:us-west-2:123456789012:my-notification-topic",
    "Id": "YmQzMmEwM2EjZWVlI0NGItNzVtZjI1MC00ZjgyLWZDBiZWNI",
    "Event": "s3:ObjectCreated:*",
    "Events": [
      "s3:ObjectCreated:*"
    ]
  }
}
```

```
    ]
  }
}
```

- Per i dettagli sull'API, consulta [GetBucketNotification AWS CLI](#) Command Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo esempio recupera la configurazione di notifica del bucket specificato

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
TopicConfigurations
```

Output:

```
Id      Topic
--      -
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Per i dettagli sull'API, vedere [GetBucketNotification](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo esempio recupera la configurazione di notifica del bucket specificato

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
TopicConfigurations
```

Output:

```
Id      Topic
--      -
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Per i dettagli sull'API, vedere [GetBucketNotification](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketPolicy** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare GetBucketPolicy.

C++

SDK per C++

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
bool AwsDoc::S3::getBucketPolicy(const Aws::String &bucketName,
                                const Aws::S3::S3ClientConfiguration
                                &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketPolicyOutcome outcome =
        s3Client.GetBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketPolicy: "
                    << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::StringStream policy_stream;
        Aws::String line;

        outcome.GetResult().GetPolicy() >> line;
        policy_stream << line;
```

```

        std::cout << "Retrieve the policy for bucket '" << bucketName << "':\n\n"
    <<
        policy_stream.str() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- Per i dettagli sull'API, [GetBucketPolicy](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Il comando seguente recupera la policy di bucket per un bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket
```

Output:

```

{
  "Policy": "{\n\"Version\": \"2008-10-17\", \"Statement\": [\n{\n\"Sid\": \"\", \"Effect\": \"Allow\", \"Principal\": \"*\", \"Action\": \"s3:GetObject\", \"Resource\": \"arn:aws:s3:::amzn-s3-demo-bucket/*\"}, {\n\"Sid\": \"\", \"Effect\": \"Deny\", \"Principal\": \"*\", \"Action\": \"s3:GetObject\", \"Resource\": \"arn:aws:s3:::amzn-s3-demo-bucket/secret/*\"}]]}"
}

```

Come ottenere e implementare una policy di bucket L'esempio seguente mostra come scaricare una policy di bucket di Amazon S3, apportare modifiche al file e quindi utilizzare `put-bucket-policy` per applicare la policy di bucket modificata. Per scaricare la policy di bucket in un file, è possibile eseguire:

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket --query Policy --output text > policy.json
```

Sarà quindi possibile modificare il file `policy.json` in base alle esigenze. Sarà infine possibile riapplicare la policy modificata al bucket S3 eseguendo il file

`policy.json`, se necessario. Sarà infine possibile riapplicare la policy modificata al bucket S3 eseguendo il file

se necessario. Sarà infine possibile riapplicare la policy modificata al bucket S3 eseguendo il file

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://  
policy.json
```

- Per i dettagli sull'API, consulta [GetBucketPolicy AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.s3.model.S3Exception;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;  
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 * <p>  
 * For more information, see the following documentation topic:  
 * <p>  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
  
public class GetBucketPolicy {  
    public static void main(String[] args) {  
        final String usage = ""
```

```

Usage:
    <bucketName>

Where:
    bucketName - The Amazon S3 bucket to get the policy from.
    "";

if (args.length != 1) {
    System.out.println(usage);
    System.exit(1);
}

String bucketName = args[0];
System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
Region region = Region.US_EAST_1;
S3Client s3 = S3Client.builder()
    .region(region)
    .build();

String polText = getPolicy(s3, bucketName);
System.out.println("Policy Text: " + polText);
s3.close();
}

/**
 * Retrieves the policy for the specified Amazon S3 bucket.
 *
 * @param s3 the {@link S3Client} instance to use for making the request
 * @param bucketName the name of the S3 bucket for which to retrieve the
policy
 * @return the policy text for the specified bucket, or an empty string if an
error occurs
 */
public static String getPolicy(S3Client s3, String bucketName) {
    String policyText;
    System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
    GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
        policyText = policyRes.policy();
    }

```

```
        return policyText;

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
}
```

- Per i dettagli sull'API, [GetBucketPolicy](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Recupera la policy del bucket.

```
import {
    GetBucketPolicyCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Logs the policy for a specified bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
    const client = new S3Client({});

    try {
        const { Policy } = await client.send(
            new GetBucketPolicyCommand({
```

```
        Bucket: bucketName,
    })),
    );
    console.log(`Policy for "${bucketName}":\n${Policy}`);
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while getting policy from ${bucketName}. The bucket
            doesn't exist.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while getting policy from ${bucketName}.  ${caught.name}:
            ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [GetBucketPolicy](#) consulta AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun getPolicy(bucketName: String): String? {
    println("Getting policy for bucket $bucketName")
```

```
val request =
    GetBucketPolicyRequest {
        bucket = bucketName
    }

S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
    val policyRes = s3.getBucketPolicy(request)
    return policyRes.policy
}
}
```

- Per i dettagli sull'API, [GetBucketPolicy](#) consulta AWS SDK for Kotlin API reference.

PowerShell

Strumenti per V4 PowerShell

Esempio 1: questo comando restituisce la policy di bucket associata al bucket S3 specificato.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [GetBucketPolicy](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando restituisce la policy di bucket associata al bucket S3 specificato.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [GetBucketPolicy](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Python

SDK per Python (Boto3)

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def get_policy(self):
        """
        Get the security policy of the bucket.

        :return: The security policy of the specified bucket, in JSON format.
        """
        try:
            policy = self.bucket.Policy()
            logger.info(
                "Got policy %s for bucket '%s'.", policy.policy, self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get policy for bucket '%s'.",
                             self.bucket.name)
            raise
        else:
            return json.loads(policy.policy)
```


- Per i dettagli sull'API, consulta [GetBucketPolicy](#) AWS SDK for Python (Boto3) API Reference.

Ruby

SDK per Ruby

Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
  # configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def policy
    policy = @bucket_policy.data.policy
    policy.respond_to?(:read) ? policy.read : policy
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
    #{e.message}"
    nil
  end
end
```

- Per i dettagli sull'API, [GetBucketPolicy](#) consulta AWS SDK per Ruby API Reference.

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.  
    oo_result = lo_s3->getbucketpolicy(          " oo_result is returned for  
testing purposes. "  
        iv_bucket = iv_bucket_name ).  
    DATA(lv_policy) = oo_result->get_policy( ).  
    MESSAGE 'Retrieved bucket policy.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Per i dettagli sulle API, [GetBucketPolicy](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketPolicyStatus** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare GetBucketPolicyStatus.

CLI

AWS CLI

Come recuperare lo stato delle policy per un bucket, che indica se il bucket è pubblico

L'esempio get-bucket-policy-status seguente recupera lo stato della policy per il bucket amzn-s3-demo-bucket.

```
aws s3api get-bucket-policy-status \
```

```
--bucket amzn-s3-demo-bucket
```

Output:

```
{
  "PolicyStatus": {
    "IsPublic": false
  }
}
```

- Per i dettagli sull'API, consulta [GetBucketPolicyStatus AWS CLI](#) Command Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando restituisce lo stato delle policy per il bucket S3 specificato, ovvero indica se il bucket è pubblico.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [GetBucketPolicyStatus](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando restituisce lo stato delle policy per il bucket S3 specificato, ovvero indica se il bucket è pubblico.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [GetBucketPolicyStatus](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketReplication** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare GetBucketReplication.

CLI

AWS CLI

Il comando seguente recupera una configurazione di replica per un bucket denominato `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-replication --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "ReplicationConfiguration": {
    "Rules": [
      {
        "Status": "Enabled",
        "Prefix": "",
        "Destination": {
          "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket-backup",
          "StorageClass": "STANDARD"
        },
        "ID": "ZmUwNzE4ZmQ4tMjVhOS00MTlkLOGI4NDkzZTIWJjNTUtYTA1"
      }
    ],
    "Role": "arn:aws:iam::123456789012:role/s3-replication-role"
  }
}
```

- Per i dettagli sull'API, consulta [GetBucketReplication AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
```

```

    * Retrieves the replication details for the specified S3 bucket.
    *
    * @param s3Client          the S3 client used to interact with the S3
service
    * @param sourceBucketName the name of the S3 bucket to retrieve the
replication details for
    *
    * @throws S3Exception if there is an error retrieving the replication
details
    */
    public static void getReplicationDetails(S3Client s3Client, String
sourceBucketName) {
        GetBucketReplicationRequest getRequest =
GetBucketReplicationRequest.builder()
            .bucket(sourceBucketName)
            .build();

        try {
            ReplicationConfiguration replicationConfig =
s3Client.getBucketReplication(getRequest).replicationConfiguration();
            ReplicationRule rule = replicationConfig.rules().get(0);
            System.out.println("Retrieved destination bucket: " +
rule.destination().bucket());
            System.out.println("Retrieved priority: " + rule.priority());
            System.out.println("Retrieved source-bucket replication rule status:
" + rule.status());

        } catch (S3Exception e) {
            System.err.println("Failed to retrieve replication details: " +
e.awsErrorDetails().errorMessage());
        }
    }
}

```

- Per i dettagli sull'API, [GetBucketReplication](#) consulta AWS SDK for Java 2.x API Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: restituisce le informazioni della configurazione di replica impostate nel bucket denominato "amzn-s3-demo-bucket".

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Per i dettagli sull'API, vedere [GetBucketReplication](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: restituisce le informazioni della configurazione di replica impostate nel bucket denominato "amzn-s3-demo-bucket".

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Per i dettagli sull'API, vedere [GetBucketReplication](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketRequestPayment** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare GetBucketRequestPayment.

CLI

AWS CLI

Come recuperare la configurazione del pagamento delle richieste per un bucket

L'esempio `get-bucket-request-payment` seguente recupera la configurazione dei pagamenti a carico del richiedente per il bucket specificato.

```
aws s3api get-bucket-request-payment \
  --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "Payer": "BucketOwner"
}
```

- Per i dettagli sull'API, consulta [GetBucketRequestPayment AWS CLI Command Reference](#).

PowerShell

Strumenti per PowerShell V4

Esempio 1: restituisce la configurazione di pagamento della richiesta per il bucket denominato "amzn-s3-demo-bucket". Per impostazione predefinita, il proprietario del bucket paga gli scaricamenti dal bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Per i dettagli sull'API, vedere [GetBucketRequestPayment](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: restituisce la configurazione di pagamento della richiesta per il bucket denominato "amzn-s3-demo-bucket". Per impostazione predefinita, il proprietario del bucket paga gli scaricamenti dal bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Per i dettagli sull'API, vedere [GetBucketRequestPayment](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketTagging** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare GetBucketTagging.

CLI

AWS CLI

Il comando seguente recupera una configurazione di tagging per un bucket denominato amzn-s3-demo-bucket.

```
aws s3api get-bucket-tagging --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "TagSet": [
    {
      "Value": "marketing",
      "Key": "organization"
    }
  ]
}
```

- Per i dettagli sull'API, consulta [GetBucketTagging AWS CLI](#) Command Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando restituisce tutti i tag associati al bucket specificato.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [GetBucketTagging](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando restituisce tutti i tag associati al bucket specificato.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [GetBucketTagging](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketVersioning** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare GetBucketVersioning.

CLI

AWS CLI

Il comando seguente recupera una configurazione di controllo delle versioni per un bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api get-bucket-versioning --bucket amzn-s3-demo-bucket
```

Output:

```
{  
  "Status": "Enabled"  
}
```

- Per i dettagli sull'API, consulta [GetBucketVersioning AWS CLI](#) Command Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando restituisce lo stato del controllo delle versioni in relazione al bucket specificato.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [GetBucketVersioning](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando restituisce lo stato del controllo delle versioni in relazione al bucket specificato.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [GetBucketVersioning](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketWebsite** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare GetBucketWebsite.

.NET

SDK per .NET

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Get the website configuration.
GetBucketWebsiteRequest getRequest = new
GetBucketWebsiteRequest()
{
    BucketName = bucketName,
};
GetBucketWebsiteResponse getResponse = await
client.GetBucketWebsiteAsync(getRequest);
Console.WriteLine($"Index document:
{getResponse.WebsiteConfiguration.IndexDocumentSuffix}");
Console.WriteLine($"Error document:
{getResponse.WebsiteConfiguration.ErrorDocument}");
```

- Per i dettagli sull'API, [GetBucketWebsite](#) consulta AWS SDK per .NET API Reference.

C++

SDK per C++

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
bool AwsDoc::S3::getWebsiteConfig(const Aws::String &bucketName,
                                   const Aws::S3::S3ClientConfiguration
                                   &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
        s3Client.GetBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();

        std::cerr << "Error: GetBucketWebsite: "
                   << err.GetMessage() << std::endl;
    } else {
        Aws::S3::Model::GetBucketWebsiteResult websiteResult =
            outcome.GetResult();

        std::cout << "Success: GetBucketWebsite: "
                   << std::endl << std::endl
                   << "For bucket '" << bucketName << "':"
                   << std::endl
                   << "Index page : "
                   << websiteResult.GetIndexDocument().GetSuffix()
                   << std::endl
                   << "Error page: "
                   << websiteResult.GetErrorDocument().GetKey()
                   << std::endl;
    }
}
```

```
    return outcome.IsSuccess();  
}
```

- Per i dettagli sull'API, [GetBucketWebsite](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Il comando seguente recupera una configurazione di siti web statici per un bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api get-bucket-website --bucket amzn-s3-demo-bucket
```

Output:

```
{  
  "IndexDocument": {  
    "Suffix": "index.html"  
  },  
  "ErrorDocument": {  
    "Key": "error.html"  
  }  
}
```

- Per i dettagli sull'API, consulta [GetBucketWebsite AWS CLI](#) Command Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Recupera la configurazione del sito web.

```
import {
  GetBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Log the website configuration for a bucket.
 * @param {{ bucketName }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetBucketWebsiteCommand({
        Bucket: bucketName,
      }),
    );
    console.log(
      `Your bucket is set up to host a website with the following configuration:
\n${JSON.stringify(response, null, 2)}\`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchWebsiteConfiguration"
    ) {
      console.error(
        `Error from S3 while getting website configuration for ${bucketName}. The
bucket isn't configured as a website.\`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting website configuration for ${bucketName}.
${caught.name}: ${caught.message}\`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Per i dettagli sull'API, [GetBucketWebsite](#) consulta AWS SDK per JavaScript API Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando restituisce i dettagli delle configurazioni dei siti web statici del bucket S3 specificato.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [GetBucketWebsite](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando restituisce i dettagli delle configurazioni dei siti web statici del bucket S3 specificato.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [GetBucketWebsite](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetObject** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare `GetObject`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- [Informazioni di base](#)
- [Recuperare un oggetto da un bucket se è stato modificato](#)
- [Ottieni un oggetto da un punto di Multi-Region accesso](#)
- [Nozioni di base sulla crittografia](#)
- [Guida introduttiva allo storage di oggetti](#)

- [Eseguire richieste condizionali](#)
- [Monitorare caricamenti e scaricamenti](#)

.NET

SDK per .NET (v4)

Note

C'è altro da fare. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>
/// Shows how to download an object from an Amazon S3 bucket to the
/// local computer.
/// </summary>
/// <param name="bucketName">The name of the bucket where the object is
/// currently stored.</param>
/// <param name="objectName">The name of the object to download.</param>
/// <param name="filePath">The path, including filename, where the
/// downloaded object will be stored.</param>
/// <returns>A boolean value indicating the success or failure of the
/// download process.</returns>
public async Task<bool> DownloadObjectFromBucketAsync(
    string bucketName,
    string objectName,
    string filePath)
{
    var request = new GetObjectRequest
    {
        BucketName = bucketName,
        Key = objectName,
    };

    using GetObjectResponse response = await
        _amazonS3.GetObjectAsync(request);

    try
    {

```

```

        // Save object to local file
        await response.WriteResponseStreamToFileAsync($"{filePath}\
\{objectName}", true, CancellationToken.None);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error saving {objectName}: {ex.Message}");
        return false;
    }
}

```

- Per i dettagli sull'API, [GetObject](#) consulta AWS SDK per .NET API Reference.

SDK per .NET

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Ottiene un oggetto utilizzando una richiesta condizionale.

```

/// <summary>
/// Retrieves an object from Amazon S3 with a conditional request.
/// </summary>
/// <param name="objectKey">The key of the object to retrieve.</param>
/// <param name="sourceBucket">The source bucket of the object.</param>
/// <param name="conditionType">The type of condition: 'IfMatch',
'IfNoneMatch', 'IfModifiedSince', 'IfUnmodifiedSince'.</param>
/// <param name="conditionDateValue">The value to use for the condition for
dates.</param>
/// <param name="etagConditionalValue">The value to use for the condition for
etags.</param>
/// <returns>True if the conditional read is successful, False otherwise.</
returns>
public async Task<bool> GetObjectConditional(string objectKey, string
sourceBucket,
    S3ConditionType conditionType, DateTime? conditionDateValue = null,
    string? etagConditionalValue = null)

```



```
{
    try
    {
        var getObjectRequest = new GetObjectRequest
        {
            BucketName = sourceBucket,
            Key = objectKey
        };

        switch (conditionType)
        {
            case S3ConditionType.IfMatch:
                getObjectRequest.EtagToMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfNoneMatch:
                getObjectRequest.EtagToNotMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfModifiedSince:
                getObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                break;
            case S3ConditionType.IfUnmodifiedSince:
                getObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                break;
            default:
                throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
        }

        var response = await _amazonS3.GetObjectAsync(getObjectRequest);
        var sampleBytes = new byte[20];
        await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
        _logger.LogInformation($"Conditional read
successful. Here are the first 20 bytes of the object:
\n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
        return true;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional read failed: Precondition failed");
        }
    }
}
```

```

        else if (e.ErrorCode == "NotModified")
        {
            _logger.LogError("Conditional read failed: Object not modified");
        }
        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return false;
    }
}

```

- Per i dettagli sull'API, [GetObject](#) consulta AWS SDK per .NET API Reference.

Bash

AWS CLI con lo script Bash

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#     $1 - The name of the bucket to download the object from.

```

```
#      $2 - The path and file name to store the downloaded bucket.
#      $3 - The key (name) of the object in the bucket.
#
# Returns:
#      0 - If successful.
#      1 - If it fails.
#####
function download_object_from_bucket() {
    local bucket_name=$1
    local destination_file_name=$2
    local object_name=$3
    local response

    response=$(aws s3api get-object \
        --bucket "$bucket_name" \
        --key "$object_name" \
        "$destination_file_name")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "ERROR: AWS reports put-object operation failed.\n$response"
        return 1
    fi
}
```

- Per i dettagli sull'API, consulta [GetObject AWS CLI](#) Command Reference.

C++

SDK per C++

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
bool AwsDoc::S3::getObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
```

```

    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome =
        client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Successfully retrieved '" << objectKey << "' from '"
            << fromBucket << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

```

- Per i dettagli sull'API, [GetObject](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

L'esempio seguente utilizza il comando `get-object` per scaricare un oggetto da Amazon S3.

```
aws s3api get-object --bucket text-content --key dir/
my_images.tar.bz2 my_images.tar.bz2
```

Nota che il parametro `outfile` è specificato senza un nome di opzione, ad esempio `--outfile`. Il nome del file di output deve essere l'ultimo parametro del comando.

L'esempio seguente mostra l'utilizzo di `--range` per scaricare un intervallo di byte specifico da un oggetto. Tieni presente che gli intervalli di byte devono essere preceduti dal prefisso `"bytes="`:

```
aws s3api get-object --bucket text-content --key dir/my_data --  
range bytes=8888-9999 my_data_range
```

Per ulteriori informazioni sul recupero degli oggetti, consulta “Recupero degli oggetti” nella Guida per gli sviluppatori di Amazon S3.

- Per i dettagli sull'API, consulta [GetObject AWS CLI Command Reference](#).

Go

SDK per Go V2

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform  
// bucket
```

```
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:    aws.String(objectKey),
    })
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
                objectKey, bucketName)
            err = noKey
        } else {
            log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
                objectKey, err)
        }
        return err
    }
    defer result.Body.Close()
    file, err := os.Create(fileName)
    if err != nil {
        log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
        return err
    }
    defer file.Close()
    body, err := io.ReadAll(result.Body)
    if err != nil {
        log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
            err)
    }
    _, err = file.Write(body)
    return err
}
```

- Per i dettagli sull'API, [GetObject](#) consulta AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Leggi i dati come array di byte utilizzando un'interfaccia [S3Client](#).

```
/**
 * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket
 * and writes them to a local file.
 *
 * @param bucketName the name of the S3 bucket containing the object
 * @param keyName     the key (or name) of the S3 object to retrieve
 * @param path        the local file path where the object's bytes will be
 * written
 * @return a {@link CompletableFuture} that completes when the object bytes
 * have been written to the local file
 */
public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
    GetObjectRequest objectRequest = GetObjectRequest.builder()
        .key(keyName)
        .bucket(bucketName)
        .build();

    CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
    return response.thenAccept(objectBytes -> {
        try {
            byte[] data = objectBytes.asByteArray();
            Path filePath = Paths.get(path);
            Files.write(filePath, data);
            logger.info("Successfully obtained bytes from an S3 object");
        } catch (IOException ex) {
```

```

        throw new RuntimeException("Failed to write data to file", ex);
    }
}).whenComplete((resp, ex) -> {
    if (ex != null) {
        throw new RuntimeException("Failed to get object bytes from S3",
ex);
    }
});
}

```

Usa un [S3TransferManager](#) per [scaricare un oggetto](#) in un bucket S3 in un file locale. Visualizza il [file completo](#) ed esegui il [test](#).

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.NoSuchFileException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String
bucketName,

                                String key, String downloadedFilePath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
            .getObjectRequest(b -> b.bucket(bucketName).key(key))
            .destination(Paths.get(downloadedFilePath))
            .build();

```



```

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }

```

Leggi i tag appartenenti a un oggetto utilizzando un'interfaccia [S3Client](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
            <bucketName> <keyName>\s

        Where:
            bucketName - The Amazon S3 bucket name.\s
            keyName - A key name that represents the object.\s
    }

```

```
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String keyName = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    listTags(s3, bucketName, keyName);
    s3.close();
}

/**
 * Lists the tags associated with an Amazon S3 object.
 *
 * @param s3 the S3Client object used to interact with the Amazon S3 service
 * @param bucketName the name of the S3 bucket that contains the object
 * @param keyName the key (name) of the S3 object
 */
public static void listTags(S3Client s3, String bucketName, String keyName) {
    try {
        GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        GetObjectTaggingResponse tags =
s3.getObjectTagging(getTaggingRequest);
        List<Tag> tagSet = tags.getTagSet();
        for (Tag tag : tagSet) {
            System.out.println(tag.getKey());
            System.out.println(tag.getValue());
        }

    } catch (S3Exception e) {
        System.err.println(e.getAwsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
    }  
  }  
}
```

Recupera un URL per un oggetto utilizzando un'interfaccia [S3Client](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.GetUrlRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
  
import java.net.URL;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 * <p>  
 * For more information, see the following documentation topic:  
 * <p>  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
  
public class GetObjectUrl {  
    public static void main(String[] args) {  
        final String usage = ""  
  
            Usage:  
            <bucketName> <keyName>\s  
  
            Where:  
            bucketName - The Amazon S3 bucket name.  
            keyName - A key name that represents the object.\s  
            "";  
  
        if (args.length != 2) {  
            System.out.println(usage);  
            System.exit(1);  
        }  
  
        String bucketName = args[0];
```

```

        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getURL(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Retrieves the URL for a specific object in an Amazon S3 bucket.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket where the object is stored
     * @param keyName the name of the object for which the URL should be
     retrieved
     * @throws S3Exception if there is an error retrieving the URL for the
     specified object
     */
    public static void getURL(S3Client s3, String bucketName, String keyName) {
        try {
            GetUrlRequest request = GetUrlRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            URL url = s3.utilities().getUrl(request);
            System.out.println("The URL for " + keyName + " is " + url);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

Recupera un oggetto utilizzando l'oggetto client S3Presigner mediante un'interfaccia [S3Client](#).

```

import java.io.IOException;
import java.io.InputStream;

```

```
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
    software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents a text file.\s
            """;

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Presigner presigner = S3Presigner.builder()
```

```

        .region(region)
        .build();

        getPresignedUrl(presigner, bucketName, keyName);
        presigner.close();
    }

    /**
     * Generates a pre-signed URL for an Amazon S3 object.
     *
     * @param presigner The {@link S3Presigner} instance to use for generating
     the pre-signed URL.
     * @param bucketName The name of the Amazon S3 bucket where the object is
     stored.
     * @param keyName The key name (file name) of the object in the Amazon S3
     bucket.
     *
     * @throws S3Exception If there is an error interacting with the Amazon S3
     service.
     * @throws IOException If there is an error opening the HTTP connection or
     reading/writing the request/response.
     */
    public static void getPresignedUrl(S3Presigner presigner, String bucketName,
    String keyName) {
        try {
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            GetObjectPresignRequest getObjectPresignRequest =
            GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(60))
                .getObjectRequest(getObjectRequest)
                .build();

            PresignedGetObjectRequest presignedGetObjectRequest =
            presigner.presignGetObject(getObjectPresignRequest);
            String theUrl = presignedGetObjectRequest.url().toString();
            System.out.println("Presigned URL: " + theUrl);
            HttpURLConnection connection = (HttpURLConnection)
            presignedGetObjectRequest.url().openConnection();
            presignedGetObjectRequest.httpRequest().headers().forEach((header,
            values) -> {

```

```

        values.forEach(value -> {
            connection.addRequestProperty(header, value);
        });
    });

    // Send any request payload that the service needs (not needed when
    // isBrowserExecutable is true).
    if (presignedGetObjectRequest.signedPayload().isPresent()) {
        connection.setDoOutput(true);

        try (InputStream signedPayload =
presignedGetObjectRequest.signedPayload().get().asInputStream();
            OutputStream httpOutputStream =
connection.getOutputStream()) {
            IoUtils.copy(signedPayload, httpOutputStream);
        }
    }

    // Download the result of executing the request.
    try (InputStream content = connection.getInputStream()) {
        System.out.println("Service returned response: ");
        IoUtils.copy(content, System.out);
    }

} catch (S3Exception | IOException e) {
    e.printStackTrace();
}

}
}

```

[Ottieni un oggetto utilizzando un ResponseTransformer oggetto e S3Client.](#)

```

import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;

import java.io.File;

```

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetObjectData {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <path>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
                path - The path where the file is written to.\s
            "";

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getObjectBytes(s3, bucketName, keyName, path);
        s3.close();
    }
}
```



```
/**
 * Retrieves the bytes of an object stored in an Amazon S3 bucket and saves
 * them to a local file.
 *
 * @param s3 The S3Client instance used to interact with the Amazon S3
 * service.
 * @param bucketName The name of the S3 bucket where the object is stored.
 * @param keyName The key (or name) of the S3 object.
 * @param path The local file path where the object's bytes will be saved.
 * @throws IOException If an I/O error occurs while writing the bytes to the
 * local file.
 * @throws S3Exception If an error occurs while retrieving the object from
 * the S3 bucket.
 */
public static void getObjectBytes(S3Client s3, String bucketName, String
keyName, String path) {
    try {
        GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        ResponseBytes<GetObjectResponse> objectBytes =
s3.getObject(objectRequest, ResponseTransformer.toBytes());
        byte[] data = objectBytes.asByteArray();

        // Write the data to a local file.
        File myFile = new File(path);
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
        System.out.println("Successfully obtained bytes from an S3 object");
        os.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, vedi [GetObject](#) in AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Scarica l'oggetto.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log(str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
```

```

        `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.` ,
    );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

```

Scarica l'oggetto a condizione che il relativo ETag corrisponda a quello fornito.

```

import {
    GetObjectCommand,
    NoSuchKey,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
    const client = new S3Client({});

    try {
        const response = await client.send(
            new GetObjectCommand({
                Bucket: bucketName,
                Key: key,
                IfMatch: eTag,
            }),
        );
        // The Body object also has 'transformToByteArray' and 'transformToWebStream'
        methods.
        const str = await response.Body.transformToString();
    }
};

```

```
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.` ,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}` ,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};
```

```

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}

```

Scarica l'oggetto a condizione che il relativo ETag non corrisponda a quello fornito.

```

import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfNoneMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(

```

```
        `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.` ,
    );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}` ,
        );
    } else {
        throw caught;
    }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
    isMain,
    validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
    const options = {
        bucketName: {
            type: "string",
            required: true,
        },
        key: {
            type: "string",
            required: true,
        },
        eTag: {
            type: "string",
            required: true,
        },
    };
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
```

```
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Scarica l'oggetto a condizione che sia stato creato o modificato in un determinato periodo di tempo.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const date = new Date();
  date.setDate(date.getDate() - 1);
  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfModifiedSince: date,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
        such key exists.`
      );
    }
  }
}
```

```
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while getting object from ${bucketName}.  ${caught.name}:
${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```


Scarica l'oggetto a condizione che non sia stato creato o modificato in un determinato periodo di tempo.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const date = new Date();
  date.setDate(date.getDate() - 1);
  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfUnmodifiedSince: date,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`,
      );
    }
  }
}
```

```
    );
  } else {
    throw caught;
  }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [GetObject](#) consulta AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun getObjectBytes(  
    bucketName: String,  
    keyName: String,  
    path: String,  
) {  
    val request =  
        GetObjectRequest {  
            key = keyName  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.getObject(request) { resp ->  
            val myFile = File(path)  
            resp.body?.writeToFile(myFile)  
            println("Successfully read $keyName from $bucketName")  
        }  
    }  
}
```

- Per i dettagli sull'API, [GetObject](#) consulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Recupera un oggetto.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $file = $this->s3client->getObject([
        'Bucket' => $this->bucketName,
        'Key' => $fileName,
    ]);
    $body = $file->get('Body');
    $body->rewind();
    echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
} catch (Exception $exception) {
    echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
    exit("Please fix error with file downloading before continuing.");
}
```

- Per i dettagli sull'API, [GetObject](#) consulta AWS SDK per PHP API Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: Questo comando recupera l'elemento "sample.txt" dal bucket «amzn-s3-demo-bucket» e lo salva in un file denominato "local-sample.txt" nella posizione corrente. Il file "local-sample.txt" non deve esistere prima di chiamare questo comando.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-
sample.txt
```

Esempio 2: questo comando recupera la directory virtuale «DIR» dal bucket «amzn-s3-demo-bucket» e la salva in una cartella denominata "" nella posizione corrente. Local-DIR La cartella "" Local-DIR non deve esistere prima che questo comando venga chiamato.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Esempio 3: scarica tutti gli oggetti con chiavi che terminano con “.json” dai bucket contenenti “config” nel relativo nome nei file nella cartella specificata. Le chiavi degli oggetti vengono utilizzate per impostare i nomi dei file.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\Config0bjects
```

- Per i dettagli sull'API, vedere [GetObject](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: Questo comando recupera l'elemento "sample.txt" dal bucket «amzn-s3-demo-bucket» e lo salva in un file denominato "local-sample.txt" nella posizione corrente. Il file “local-sample.txt” non deve esistere prima di chiamare questo comando.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

Esempio 2: questo comando recupera la directory virtuale «DIR» dal bucket «amzn-s3-demo-bucket» e la salva in una cartella denominata "" nella posizione corrente. Local-DIR La cartella "" Local-DIR non deve esistere prima che questo comando venga chiamato.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Esempio 3: scarica tutti gli oggetti con chiavi che terminano con “.json” dai bucket contenenti “config” nel relativo nome nei file nella cartella specificata. Le chiavi degli oggetti vengono utilizzate per impostare i nomi dei file.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\Config0bjects
```

Esempio 4: scarica un singolo oggetto utilizzando il download parallelo multiparte per migliorare le prestazioni su file di grandi dimensioni.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt -UseMultipartDownload
```

Esempio 5: scarica un oggetto di grandi dimensioni utilizzando la modalità RANGE con una dimensione della parte personalizzata di 64 MB. La modalità RANGE utilizza richieste HTTP con intervallo di byte e funziona per tutti gli oggetti indipendentemente dal modo in cui sono stati caricati.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -PartSize 64MB
```

Esempio 6: scarica tutti gli oggetti con il prefisso chiave «data/» in una cartella locale utilizzando il download multipart con download simultanei di file.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-data -UseMultipartDownload -DownloadFilesConcurrently
```

Esempio 7: scarica un singolo oggetto utilizzando il download in più parti con 20 connessioni simultanee per la massima velocità di trasmissione su reti a larghezza di banda elevata.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

Esempio 8: scarica tutti gli oggetti con il prefisso chiave «data/» in una cartella locale utilizzando il download in più parti, continuando a superare gli errori dei singoli file anziché interrompere.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- Per i dettagli sull'API, vedere [GetObject](#) in Cmdlet Reference (V5).AWS Strumenti per PowerShell

Python

SDK per Python (Boto3)

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def get(self):
        """
        Gets the object.

        :return: The object data in bytes.
        """
        try:
            body = self.object.get()["Body"].read()
            logger.info(
                "Got object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't get object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
```

```
        raise
    else:
        return body
```

Ottiene un oggetto utilizzando una richiesta condizionale.

```
class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def get_object_conditional(
        self,
        object_key: str,
        source_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Retrieves an object from Amazon S3 with a conditional request.

        :param object_key: The key of the object to retrieve.
        :param source_bucket: The source bucket of the object.
        :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
        'IfModifiedSince', 'IfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            response = self.s3.get_object(
                Bucket=source_bucket,
```



```

        Key=object_key,
        **{condition_type: condition_value},
    )
    sample_bytes = response["Body"].read(20)
    print(
        f"\tConditional read successful. Here are the first 20 bytes of
the object:\n"
    )
    print(f"\t{sample_bytes}")
except ClientError as e:
    error_code = e.response["Error"]["Code"]
    if error_code == "PreconditionFailed":
        print("\tConditional read failed: Precondition failed")
    elif error_code == "304": # Not modified error code.
        print("\tConditional read failed: Object not modified")
    else:
        logger.error(f"Unexpected error: {error_code}")
        raise

```

- Per i dettagli sull'API, consulta [GetObject AWS SDK for Python \(Boto3\) API Reference](#).

Ruby

SDK per Ruby

Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Recupera un oggetto.

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.

```

```

def initialize(object)
  @object = object
end

# Gets the object directly to a file.
#
# @param target_path [String] The path to the file where the object is
downloaded.
# @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
successful; otherwise nil.
def get_object(target_path)
  @object.get(response_target: target_path)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

  wrapper = ObjectGetWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  obj_data = wrapper.get_object(target_path)
  return unless obj_data

  puts "Object #{object_key} (#{obj_data.content_length} bytes) downloaded to
#{target_path}."
end

run_demo if $PROGRAM_NAME == __FILE__

```

Recupera un oggetto e segnala lo stato di crittografia lato server.

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.

```

```
def initialize(object)
  @object = object
end

# Gets the object into memory.
#
# @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
successful; otherwise nil.
def object
  @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? 'no' :
obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, [GetObject](#) consulta AWS SDK per Ruby API Reference.

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
async fn get_object(client: Client, opt: Opt) -> Result<usize, S3ExampleError> {
    trace!("bucket:      {}", opt.bucket);
    trace!("object:      {}", opt.object);
    trace!("destination: {}", opt.destination.display());

    let mut file = File::create(opt.destination.clone()).map_err(|err| {
        S3ExampleError::new(format!(
            "Failed to initialize file for saving S3 download: {err:?}"
        ))
    })?;

    let mut object = client
        .get_object()
        .bucket(opt.bucket)
        .key(opt.object)
        .send()
        .await?;

    let mut byte_count = 0_usize;
    while let Some(bytes) = object.body.try_next().await.map_err(|err| {
        S3ExampleError::new(format!("Failed to read from S3 download stream:
{err:?}"))
    })? {
        let bytes_len = bytes.len();
        file.write_all(&bytes).map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to write from S3 download stream to local file: {err:?}"
            ))
        })?;
        trace!("Intermediate write of {bytes_len}");
        byte_count += bytes_len;
    }
}
```

```
    Ok(byte_count)
}
```

- Per i dettagli sulle API, consulta la [GetObject](#) guida di riferimento all'API AWS SDK for Rust.

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.
    oo_result = lo_s3->getobject(          " oo_result is returned for
testing purposes. "
        iv_bucket = iv_bucket_name
        iv_key = iv_object_key ).
    DATA(lv_object_data) = oo_result->get_body( ).
    MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.
```

- Per i dettagli sulle API, [GetObject](#) consulta AWS SDK for SAP ABAP API reference.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import AWSS3

public func downloadFile(bucket: String, key: String, to: String) async
throws {
    let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)

    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )
    do {
        let output = try await client.getObject(input: input)

        guard let body = output.body else {
            throw HandlerError.getObjectBody("GetObjectInput missing body.")
        }

        guard let data = try await body.readData() else {
            throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
        }

        try data.write(to: fileUrl)
    }
    catch {
        print("ERROR: ", dump(error, name: "Downloading a file. "))
        throw error
    }
}
```

```
import AWSS3
```

```
public func readFile(bucket: String, key: String) async throws -> Data {
    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )
    do {
        let output = try await client.getObject(input: input)

        guard let body = output.body else {
            throw HandlerError.getObjectBody("GetObjectInput missing body.")
        }

        guard let data = try await body.readData() else {
            throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
        }

        return data
    }
    catch {
        print("ERROR: ", dump(error, name: "Reading a file. "))
        throw error
    }
}
```

- Per i dettagli sull'API, consulta la [GetObject](#) guida di riferimento all'API AWS SDK for Swift.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetObjectAcl** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare `GetObjectAcl`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. Puoi vedere questa azione nel contesto nel seguente esempio di codice:

- [Gestire le liste di controllo degli accessi \(ACL\)](#)

C++

SDK per C++

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
bool AwsDoc::S3::getObjectAcl(const Aws::String &bucketName,
                              const Aws::String &objectKey,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetObjectAclRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectAclOutcome outcome =
        s3Client.GetObjectAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObjectAcl: "
                    << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
            outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            std::cout << "For object " << objectKey << ": "
                        << std::endl << std::endl;

            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
```



```

        << getGranteeTypeString(grantee.GetType()) <<
std::endl;
    }

    if (grantee.DisplayNameHasBeenSet()) {
        std::cout << "Display name: "
        << grantee.GetDisplayName() << std::endl;
    }

    if (grantee.EmailAddressHasBeenSet()) {
        std::cout << "Email address: "
        << grantee.GetEmailAddress() << std::endl;
    }

    if (grantee.IDHasBeenSet()) {
        std::cout << "ID: "
        << grantee.GetID() << std::endl;
    }

    if (grantee.URIHasBeenSet()) {
        std::cout << "URI: "
        << grantee.GetURI() << std::endl;
    }

    std::cout << "Permission: " <<
        getPermissionString(grant.GetPermission()) <<
        std::endl << std::endl;
    }
}

return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string
 */
Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:

```

```

        return "Canonical user ID of an AWS account";
    case Aws::S3::Model::Type::Group:
        return "Predefined Amazon S3 group";
    case Aws::S3::Model::Type::NOT_SET:
        return "Not set";
    default:
        return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
    string.
    /*!
    \param permission: Permission enumeration.
    \return String: Human-readable string
    */
    Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
        switch (permission) {
            case Aws::S3::Model::Permission::FULL_CONTROL:
                return "Can read this object's data and its metadata, "
                    "and read/write this object's permissions";
            case Aws::S3::Model::Permission::NOT_SET:
                return "Permission not set";
            case Aws::S3::Model::Permission::READ:
                return "Can read this object's data and its metadata";
            case Aws::S3::Model::Permission::READ_ACP:
                return "Can read this object's permissions";
            // case Aws::S3::Model::Permission::WRITE // Not applicable.
            case Aws::S3::Model::Permission::WRITE_ACP:
                return "Can write this object's permissions";
            default:
                return "Permission unknown";
        }
    }
}

```

- Per i dettagli sull'API, [GetObjectAcl](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Il comando seguente recupera una lista di controllo degli accessi (ACL) per un oggetto in un bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api get-object-acl --bucket amzn-s3-demo-bucket --key index.html
```

Output:

```
{
  "Owner": {
    "DisplayName": "my-username",
    "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
  },
  "Grants": [
    {
      "Grantee": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
      },
      "Permission": "FULL_CONTROL"
    },
    {
      "Grantee": {
        "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
      },
      "Permission": "READ"
    }
  ]
}
```

- Per i dettagli sull'API, consulta [GetObjectAcl AWS CLI Command Reference](#).

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- Per i dettagli sull'API, [GetObjectAcl](#) consulta AWS SDK for Kotlin API reference.

PowerShell

Strumenti per V5 PowerShell

Esempio 1: questo comando ottiene i dettagli del proprietario dell'oggetto S3.

```
(Get-S3ObjectACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select
*).Owner
```

Output:

```

DisplayName Id
----- --
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544

```

- Per i dettagli sull'API, vedere [GetObjectAcl](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Python

SDK per Python (Boto3)

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                               that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def get_acl(self):
        """
        Gets the ACL of the object.

        :return: The ACL of the object.
        """
        try:
            acl = self.object.Acl()
            logger.info(
                "Got ACL for object %s owned by %s.",

```

```

        self.object.key,
        acl.owner["DisplayName"],
    )
except ClientError:
    logger.exception("Couldn't get ACL for object %s.", self.object.key)
    raise
else:
    return acl

```

- Per i dettagli sull'API, consulta [GetObjectAcl AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

TRY.
    oo_result = lo_s3->getobjectacl(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name
    iv_key = iv_object_key ).
    MESSAGE 'Retrieved object ACL.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

```

- Per i dettagli sulle API, [GetObjectAcl](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetObjectAttributes** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare `GetObjectAttributes`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. Puoi vedere questa azione nel contesto nel seguente esempio di codice:

- [Utilizzare la funzionalità per l'integrità degli oggetti di Amazon S3](#)

C++

SDK per C++

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object is stored.
    \param key: The unique identifier (key) of the object within the S3 bucket.
    \param hashMethod: The hashing algorithm used to calculate the hash value of
the object.
    \param[out] hashData: The retrieved hash.
    \param[out] partHashes: The part hashes if available.
    \param client: The S3 client instance used to retrieve the object.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
&key,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   Aws::String &hashData,
                                   std::vector<Aws::String> *partHashes,
                                   const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
```

```

request.SetBucket(bucket);
request.SetKey(key);

if (hashMethod == MD5) {
    Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
    attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
    request.SetObjectAttributes(attributes);

    Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        hashData = result.GetETag();
    } else {
        std::cerr << "Error retrieving object etag attributes." <<
outcome.GetError().GetMessage() << std::endl;
        return false;
    }
} else { // hashMethod != MD5
    Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
    attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
    request.SetObjectAttributes(attributes);

    Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        switch (hashMethod) {
            case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                break; // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
            case AwsDoc::S3::MD5:
                break; // MD5 is not supported.
#pragma clang diagnostic pop
            case AwsDoc::S3::SHA1:
                hashData = result.GetChecksum().GetChecksumSHA1();
                break;
            case AwsDoc::S3::SHA256:
                hashData = result.GetChecksum().GetChecksumSHA256();

```



```

        break;
    case AwsDoc::S3::CRC32:
        hashData = result.GetChecksum().GetChecksumCRC32();
        break;
    case AwsDoc::S3::CRC32C:
        hashData = result.GetChecksum().GetChecksumCRC32C();
        break;
    default:
        std::cerr << "Unknown hash method." << std::endl;
        return false;
    }
} else {
    std::cerr << "Error retrieving object checksum attributes." <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

if (nullptr != partHashes) {
    attributes.clear();
    attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
    request.SetObjectAttributes(attributes);
    outcome = client.GetObjectAttributes(request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
        for (const Aws::S3::Model::ObjectPart &part: parts) {
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                    break;
                case AwsDoc::S3::MD5: // MD5 is not supported.
                    break;
                case AwsDoc::S3::SHA1:
                    partHashes->push_back(part.GetChecksumSHA1());
                    break;
                case AwsDoc::S3::SHA256:
                    partHashes->push_back(part.GetChecksumSHA256());
                    break;
                case AwsDoc::S3::CRC32:
                    partHashes->push_back(part.GetChecksumCRC32());
                    break;
                case AwsDoc::S3::CRC32C:

```

```

        partHashes->push_back(part.GetChecksumCRC32C());
        break;
    default:
        std::cerr << "Unknown hash method." << std::endl;
        return false;
    }
}
} else {
    std::cerr << "Error retrieving object attributes for object
parts." <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}
}
}
return true;
}

```

- Per i dettagli sull'API, [GetObjectAttributes](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Come recuperare i metadati da un oggetto senza restituire l'oggetto stesso

L'esempio `get-object-attributes` seguente recupera i metadati dall'oggetto `doc1.rtf`.

```

aws s3api get-object-attributes \
  --bucket amzn-s3-demo-bucket \
  --key doc1.rtf \
  --object-attributes "StorageClass" "ETag" "ObjectSize"

```

Output:

```

{
  "LastModified": "2022-03-15T19:37:31+00:00",
  "VersionId": "IuCPjXTDzHNfIdAuitVBIKJpF2p1fg4P",
  "ETag": "b662d79adeb7c8d787ea7eafb9ef6207",
  "StorageClass": "STANDARD",

```

```
"ObjectSize": 405
}
```

Per ulteriori informazioni, [GetObjectAttributes](#) consulta Amazon S3 API Reference.

- Per i dettagli sull'API, consulta [GetObjectAttributes AWS CLI](#) Command Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetObjectLegalHold** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare GetObjectLegalHold.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Bloccare gli oggetti di Amazon S3](#)

.NET

SDK per .NET

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>
/// Get the legal hold details for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object legal hold details.</returns>
public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
    string objectKey)
{
    try
```

```
{
    var request = new GetObjectLegalHoldRequest()
    {
        BucketName = bucketName,
        Key = objectKey
    };

    var response = await _amazonS3.GetObjectLegalHoldAsync(request);
    Console.WriteLine($"\\tObject legal hold for {objectKey} in
{bucketName}: " +
                    $"\\n\\tStatus: {response.LegalHold.Status}");
    return response.LegalHold;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"\\tUnable to fetch legal hold: '{ex.Message}'");
    return new ObjectLockLegalHold();
}
}
```

- Per i dettagli sull'API, [GetObjectLegalHold](#) consulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Come recuperare lo stato relativo alla conservazione a fini legali di un oggetto

L'esempio `get-object-legal-hold` seguente recupera lo stato relativo alla conservazione a fini legali per l'oggetto specificato.

```
aws s3api get-object-legal-hold \
  --bucket amzn-s3-demo-bucket-with-object-lock \
  --key doc1.rtf
```

Output:

```
{
  "LegalHold": {
    "Status": "ON"
  }
}
```

```
}
```

- Per i dettagli sull'API, consulta [GetObjectLegalHold AWS CLI Command Reference](#).

Go

SDK per Go V2

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// GetObjectLegalHold retrieves the legal hold status for an S3 object.  
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key  
    string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {  
    var status *types.ObjectLockLegalHoldStatus
```

```
input := &s3.GetObjectLegalHoldInput{
    Bucket:    aws.String(bucket),
    Key:       aws.String(key),
    VersionId: aws.String(versionId),
}

output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
if err != nil {
    var noSuchKeyErr *types.NoSuchKey
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noSuchKeyErr) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noSuchKeyErr
    } else if errors.As(err, &apiErr) {
        switch apiErr.ErrorCode() {
        case "NoSuchObjectLockConfiguration":
            log.Printf("Object %s does not have an object lock configuration.\n", key)
            err = nil
        case "InvalidRequest":
            log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
            err = nil
        }
    }
} else {
    status = &output.LegalHold.Status
}

return status, err
}
```

- Per i dettagli sull'API, [GetObjectLegalHold](#) consulta AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
        GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
        System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
            ":\n\tStatus: " + response.legalHold().status());
        return response.legalHold();

    } catch (S3Exception ex) {
        System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
    }

    return null;
}
```

- Per i dettagli sull'API, [GetObjectLegalHold](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import {
    GetObjectLegalHoldCommand,
```

```
S3Client,
S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get an object's current legal hold status.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectLegalHoldCommand({
        Bucket: bucketName,
        Key: key,
        // Optionally, you can provide additional parameters
        // ExpectedBucketOwner: "<account ID that is expected to own the
bucket>",
        // VersionId: "<the specific version id of the object to check>",
      }),
    );
    console.log(`Legal Hold Status: ${response.LegalHold.Status}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting legal hold status for ${key} in
${bucketName}. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting legal hold status for ${key} in
${bucketName} from ${bucketName}. ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
```



```
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Per i dettagli sull'API, [GetObjectLegalHold](#) consulta AWS SDK per JavaScript API Reference.

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Applica a un oggetto lo stato relativo alla conservazione a fini legali.

```
def get_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Get the legal hold status of a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to get the legal hold status of.
    """
    print()
    logger.info("Getting legal hold status of file [%s] in bucket [%s]", key,
                bucket)
    try:
        response = s3_client.get_object_legal_hold(Bucket=bucket, Key=key)
        legal_hold_status = response["LegalHold"]["Status"]
        logger.debug(
            "Legal hold status of file [%s] in bucket [%s] is [%s]",
            key,
            bucket,
            legal_hold_status,
        )
    except Exception as e:
        logger.error(
            "Failed to get legal hold status of file [%s] in bucket [%s]: %s",
            key,
            bucket,
            e,
        )
```

- Per i dettagli sull'API, consulta [GetObjectLegalHold AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.  
    oo_result = lo_s3->getobjectlegalhold(          " oo_result is returned  
for testing purposes. "  
    iv_bucket = iv_bucket_name  
    iv_key = iv_object_key ).  
    MESSAGE 'Retrieved object legal hold status.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
CATCH /aws1/cx_s3_nosuchkey.  
    MESSAGE 'Object key does not exist.' TYPE 'E'.  
ENDTRY.
```

- Per i dettagli sulle API, [GetObjectLegalHold](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetObjectLockConfiguration** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare GetObjectLockConfiguration.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Bloccare gli oggetti di Amazon S3](#)

.NET

SDK per .NET

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>
/// Get the object lock configuration details for an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket to get details.</param>
/// <returns>The bucket's object lock configuration details.</returns>
public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
{
    try
    {
        var request = new GetObjectLockConfigurationRequest()
        {
            BucketName = bucketName
        };

        var response = await
_amazonS3.GetObjectLockConfigurationAsync(request);
        Console.WriteLine($"\\tBucket object lock config for {bucketName} in
{bucketName}: " +
                        $"\\n\\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                        $"\\n\\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

        return response.ObjectLockConfiguration;
    }
}
```

```
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tUnable to fetch object lock config:
'{ex.Message}'");
            return new ObjectLockConfiguration();
        }
    }
```

- Per i dettagli sull'API, [GetObjectLockConfiguration](#) consulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Come recupera la configurazione Object Lock di un bucket

L'esempio `get-object-lock-configuration` seguente recupera la configurazione di Object Lock per il bucket specificato.

```
aws s3api get-object-lock-configuration \
    --bucket amzn-s3-demo-bucket-with-object-lock
```

Output:

```
{
  "ObjectLockConfiguration": {
    "ObjectLockEnabled": "Enabled",
    "Rule": {
      "DefaultRetention": {
        "Mode": "COMPLIANCE",
        "Days": 50
      }
    }
  }
}
```

- Per i dettagli sull'API, consulta [GetObjectLockConfiguration AWS CLI](#) Command Reference.

Go

SDK per Go V2

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// GetObjectLockConfiguration retrieves the object lock configuration for an S3  
// bucket.  
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket  
    string) (*types.ObjectLockConfiguration, error) {  
    var lockConfig *types.ObjectLockConfiguration  
    input := &s3.GetObjectLockConfigurationInput{  
        Bucket: aws.String(bucket),  
    }  
}
```

```
output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
if err != nil {
    var noBucket *types.NoSuchBucket
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
"ObjectLockConfigurationNotFoundError" {
        log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
        err = nil
    }
} else {
    lockConfig = output.ObjectLockConfiguration
}

return lockConfig, err
}
```

- Per i dettagli sull'API, [GetObjectLockConfiguration](#) consulta AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Get the object lock configuration details for an S3 bucket.
public void getBucketObjectLockConfiguration(String bucketName) {
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .build();
```

```
GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
System.out.println("Bucket object lock config for "+bucketName+": ");
System.out.println("\tEnabled:
"+response.getObjectLockConfiguration().objectLockEnabled());
System.out.println("\tRule: "+
response.getObjectLockConfiguration().rule().defaultRetention());
}
```

- Per i dettagli sull'API, [GetObjectLockConfiguration](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import {
  GetObjectLockConfigurationCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Gets the Object Lock configuration for a bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { ObjectLockConfiguration } = await client.send(
      new GetObjectLockConfigurationCommand({
        Bucket: bucketName,
```



```

        // Optionally, you can provide additional parameters
        // ExpectedBucketOwner: "<account ID that is expected to own the
bucket>",
    }),
);
console.log(
    `Object Lock Configuration:\n${JSON.stringify(ObjectLockConfiguration)}`,
);
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while getting object lock configuration for ${bucketName}.
The bucket doesn't exist.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while getting object lock configuration for ${bucketName}.
${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
    isMain,
    validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
    const options = {
        bucketName: {
            type: "string",
            required: true,
        },
    };
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);

```

```
    return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
        main(results.values);
    } else {
        console.error(errors.join("\n"));
    }
}
```

- Per i dettagli sull'API, [GetObjectLockConfiguration](#) consulta AWS SDK per JavaScript API Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando restituisce il valore “Enabled” se la configurazione Object lock è abilitata per il bucket S3 specificato.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
ObjectLockConfiguration.ObjectLockEnabled
```

Output:

```
Value
-----
Enabled
```

- Per i dettagli sull'API, vedere [GetObjectLockConfiguration](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando restituisce il valore “Enabled” se la configurazione Object lock è abilitata per il bucket S3 specificato.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select  
ObjectLockConfiguration.ObjectLockEnabled
```

Output:

```
Value  
-----  
Enabled
```

- Per i dettagli sull'API, vedere [GetObjectLockConfiguration](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Python

SDK per Python (Boto3)

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Ottiene la configurazione Object Lock.

```
def is_object_lock_enabled(s3_client, bucket: str) -> bool:  
    """  
    Check if object lock is enabled for a bucket.  
  
    Args:  
        s3_client: Boto3 S3 client.  
        bucket: The name of the bucket to check.  
  
    Returns:  
        True if object lock is enabled, False otherwise.  
    """  
    try:  
        response = s3_client.get_object_lock_configuration(Bucket=bucket)  
        return (  
            "ObjectLockConfiguration" in response  
            and response["ObjectLockConfiguration"]["ObjectLockEnabled"] ==  
            "Enabled"
```

```

    )
except s3_client.exceptions.ClientError as e:
    if e.response["Error"]["Code"] == "ObjectLockConfigurationNotFoundError":
        return False
    else:
        raise

```

- Per i dettagli sull'API, consulta [GetObjectLockConfiguration AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

TRY.
    oo_result = lo_s3->getobjectlockconfiguration(      " oo_result is
returned for testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved object lock configuration.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- Per i dettagli sulle API, [GetObjectLockConfiguration](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetObjectRetention** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare `GetObjectRetention`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Bloccare gli oggetti di Amazon S3](#)

.NET

SDK per .NET

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>
/// Get the retention period for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object retention details.</returns>
public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
    string objectKey)
{
    try
    {
        var request = new GetObjectRetentionRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };

        var response = await _amazonS3.GetObjectRetentionAsync(request);
        Console.WriteLine($"Object retention for {objectKey} in
{bucketName}: " +
                        $"{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}");
    }
}
```

```
        return response.Retention;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to fetch object lock retention:
'{ex.Message}');
        return new ObjectLockRetention();
    }
}
```

- Per i dettagli sull'API, [GetObjectRetention](#) consulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Come recuperare la configurazione di conservazione degli oggetti per un oggetto

L'esempio `get-object-retention` seguente recupera la configurazione di conservazione degli oggetti per il bucket specificato.

```
aws s3api get-object-retention \
  --bucket amzn-s3-demo-bucket-with-object-lock \
  --key doc1.rtf
```

Output:

```
{
  "Retention": {
    "Mode": "GOVERNANCE",
    "RetainUntilDate": "2025-01-01T00:00:00.000Z"
  }
}
```

- Per i dettagli sull'API, consulta [GetObjectRetention AWS CLI](#) Command Reference.

Go

SDK per Go V2

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// GetObjectRetention retrieves the object retention configuration for an S3  
// object.  
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key  
string) (*types.ObjectLockRetention, error) {  
    var retention *types.ObjectLockRetention  
    input := &s3.GetObjectRetentionInput{  
        Bucket: aws.String(bucket),  
        Key:    aws.String(key),
```

```

}

output, err := actor.S3Client.GetObjectRetention(ctx, input)
if err != nil {
    var noKey *types.NoSuchKey
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noKey) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noKey
    } else if errors.As(err, &apiErr) {
        switch apiErr.ErrorCode() {
        case "NoSuchObjectLockConfiguration":
            err = nil
        case "InvalidRequest":
            log.Printf("Bucket %s does not have locking enabled.", bucket)
            err = nil
        }
    }
} else {
    retention = output.Retention
}

return retention, err
}

```

- Per i dettagli sull'API, [GetObjectRetention](#) consulta AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

// Get the retention period for an S3 object.
public ObjectLockRetention getObjectRetention(String bucketName, String key){
    try {

```



```

        GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

        GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
        System.out.println("Object retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}

```

- Per i dettagli sull'API, [GetObjectRetention](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import {
    GetObjectRetentionCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Log the "RetainUntilDate" for an object in an S3 bucket.
 * @param {{ bucketName: string, key: string }}

```

```

*/
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const { Retention } = await client.send(
      new GetObjectRetentionCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    console.log(
      `${key} in ${bucketName} will be retained until
      ${Retention.RetainUntilDate}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchObjectLockConfiguration"
    ) {
      console.warn(
        `The object "${key}" in the bucket "${bucketName}" does not have an
        ObjectLock configuration.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object retention settings for
        "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {

```

```
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Per i dettagli sull'API, [GetObjectRetention](#) consulta AWS SDK per JavaScript API Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: il comando restituisce la modalità e la data relative alla conservazione dell'oggetto.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- Per i dettagli sull'API, vedere [GetObjectRetention](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: il comando restituisce la modalità e la data relative alla conservazione dell'oggetto.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- Per i dettagli sull'API, vedere [GetObjectRetention](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetObjectTagging** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare GetObjectTagging.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Nozioni di base sui tag](#)

CLI

AWS CLI

Come recuperare i tag associati a un oggetto

L'esempio get-object-tagging seguente recupera i valori della chiave specificata dall'oggetto specificato.

```
aws s3api get-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc1.rtf
```

Output:

```
{  
  "TagSet": [  
    {  
      "Value": "confidential",  
      "Key": "designation"  
    }  
  ]  
}
```

L'esempio `get-object-tagging` seguente tenta di recuperare i set di tag dell'oggetto `doc2.rtf`, che non ha tag.

```
aws s3api get-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc2.rtf
```

Output:

```
{  
  "TagSet": []  
}
```

L'esempio `get-object-tagging` seguente recupera i set di tag dell'oggetto `doc3.rtf`, che ha più tag.

```
aws s3api get-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc3.rtf
```

Output:

```
{  
  "TagSet": [  
    {  
      "Value": "confidential",  
      "Key": "designation"  
    },  
    {  
      "Value": "finance",  
      "Key": "department"  
    },  
    {  
      "Value": "payroll",  
      "Key": "team"  
    }  
  ]  
}
```

- Per i dettagli sull'API, consulta [GetObjectTagging AWS CLI Command Reference](#).

PowerShell

Strumenti per PowerShell V4

Esempio 1: l'esempio restituisce i tag associati all'oggetto presente nel bucket S3 specificato.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Key  Value
---  -
test value
```

- Per i dettagli sull'API, vedere [GetObjectTagging](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: l'esempio restituisce i tag associati all'oggetto presente nel bucket S3 specificato.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Key  Value
---  -
test value
```

- Per i dettagli sull'API, vedere [GetObjectTagging](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetPublicAccessBlock** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare GetPublicAccessBlock.

CLI

AWS CLI

Come impostare o modificare la configurazione di blocco dell'accesso pubblico per un bucket

L'esempio `get-public-access-block` seguente visualizza la configurazione del blocco dell'accesso pubblico per il bucket specificato.

```
aws s3api get-public-access-block \
  --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "PublicAccessBlockConfiguration": {
    "IgnorePublicAcls": true,
    "BlockPublicPolicy": true,
    "BlockPublicAcls": true,
    "RestrictPublicBuckets": true
  }
}
```

- Per i dettagli sull'API, consulta [GetPublicAccessBlock AWS CLI Command Reference](#).

PowerShell

Strumenti per PowerShell V4

Esempio 1: il comando restituisce la configurazione del blocco dell'accesso pubblico del bucket S3 specificato.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [GetPublicAccessBlock](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: il comando restituisce la configurazione del blocco dell'accesso pubblico del bucket S3 specificato.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [GetPublicAccessBlock](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **HeadBucket** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare HeadBucket.

Bash

AWS CLI con lo script Bash

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
#####
# function bucket_exists
#
# This function checks to see if the specified bucket already exists.
#
# Parameters:
#     $1 - The name of the bucket to check.
#
# Returns:
#     0 - If the bucket already exists.
#     1 - If the bucket doesn't exist.
#####
function bucket_exists() {
    local bucket_name
    bucket_name=$1

    # Check whether the bucket already exists.
    # We suppress all output - we're interested only in the return code.
```



```
if aws s3api head-bucket \  
  --bucket "$bucket_name" \  
  >/dev/null 2>&1; then  
  return 0 # 0 in Bash script means true.  
else  
  return 1 # 1 in Bash script means false.  
fi  
}
```

- Per i dettagli sull'API, consulta [HeadBucket AWS CLI](#) Command Reference.

CLI

AWS CLI

Il comando seguente verifica l'accesso a un bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api head-bucket --bucket amzn-s3-demo-bucket
```

Se il bucket esiste e si dispone dell'accesso a esso, non viene restituito alcun output. In caso contrario, verrà visualizzato un messaggio di errore. Ad esempio:

```
A client error (404) occurred when calling the HeadBucket operation: Not Found
```

- Per i dettagli sull'API, consulta [HeadBucket AWS CLI](#) Command Reference.

Go

SDK per Go V2

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (
```

```
"bytes"
"context"
"errors"
"fmt"
"io"
"log"
"os"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
    (bool, error) {
    _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
        Bucket: aws.String(bucketName),
    })
    exists := true
    if err != nil {
        var apiError smithy.APIError
        if errors.As(err, &apiError) {
            switch apiError.(type) {
            case *types.NotFound:
                log.Printf("Bucket %v is available.\n", bucketName)
                exists = false
                err = nil
            default:
```

```
    log.Printf("Either you don't have access to bucket %v or another error
occurred. "+
    "Here's what happened: %v\n", bucketName, err)
}
}
} else {
    log.Printf("Bucket %v exists and you already own it.", bucketName)
}

return exists, err
}
```

- Per i dettagli sull'API, [HeadBucket](#) consulta AWS SDK per Go API Reference.

PowerShell

Strumenti per PowerShell V5

Esempio 1: questo comando restituisce l'output con il codice di stato HTTP 200 OK per il bucket esistente quando l'utente dispone del permesso di accedervi. BucketArn il parametro è supportato solo per i bucket di directory S3.

```
Get-S3HeadBucket -BucketName amzn-s3-demo-bucket
```

Output:

```
AccessPointAlias    : False
BucketArn           :
BucketLocationName  :
BucketLocationType  :
BucketRegion        : us-east-2
ResponseMetadata    : Amazon.Runtime.ResponseMetadata
ContentLength       : 0
HttpStatusCode       : OK
```

Esempio 2: questo comando genera un errore con il codice di stato HTTP NotFound per un bucket inesistente.

```
Get-S3HeadBucket -BucketName amzn-s3-non-existing-bucket
```

Output:

```
Get-S3HeadBucket: Error making request with Error Code NotFound and Http Status Code NotFound. No further error information was returned by the service.
```

Esempio 3: questo comando genera un errore con il codice di stato HTTP Forbidden per il bucket esistente a cui l'utente non è autorizzato ad accedere.

```
Get-S3HeadBucket -BucketName amzn-s3-no-access-bucket
```

Output:

```
Get-S3HeadBucket: Error making request with Error Code Forbidden and Http Status Code Forbidden. No further error information was returned by the service.
```

- Per i dettagli sull'API, vedere [HeadBucket](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Python

SDK per Python (Boto3)

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name
```

```
def exists(self):
    """
    Determine whether the bucket exists and you have access to it.

    :return: True when the bucket exists; otherwise, False.
    """
    try:
        self.bucket.meta.client.head_bucket(Bucket=self.bucket.name)
        logger.info("Bucket %s exists.", self.bucket.name)
        exists = True
    except ClientError:
        logger.warning(
            "Bucket %s doesn't exist or you don't have access to it.",
            self.bucket.name,
        )
        exists = False
    return exists
```

- Per i dettagli sull'API, consulta [HeadBucket AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.
    oo_result = lo_s3->headbucket(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Bucket exists and you have access to it.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Per i dettagli sulle API, [HeadBucket](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **HeadObject** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare HeadObject.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- [Controlla se esiste un oggetto](#)
- [Guida introduttiva allo storage di oggetti](#)

CLI

AWS CLI

Il comando seguente recupera i metadati di un oggetto in un bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html
```

Output:

```
{
  "AcceptRanges": "bytes",
  "ContentType": "text/html",
  "LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
  "ContentLength": 77,
  "VersionId": "null",
  "ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
  "Metadata": {}
}
```

- Per i dettagli sull'API, consulta [HeadObject AWS CLI](#) Command Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Determinazione del tipo di contenuto di un oggetto.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
            "";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String bucketName = args[0];
    String keyName = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    getContentType(s3, bucketName, keyName);
    s3.close();
}

/**
 * Retrieves the content type of an object stored in an Amazon S3 bucket.
 *
 * @param s3 an instance of the {@link S3Client} class, which is used to
interact with the Amazon S3 service
 * @param bucketName the name of the S3 bucket where the object is stored
 * @param keyName the key (file name) of the object in the S3 bucket
 */
public static void getContentType(S3Client s3, String bucketName, String
keyName) {
    try {
        HeadObjectRequest objectRequest = HeadObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        HeadObjectResponse objectHead = s3.headObject(objectRequest);
        String type = objectHead.contentType();
        System.out.println("The object content type is " + type);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

Controllo dello stato di ripristino di un oggetto.


```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        checkStatus(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Checks the restoration status of an Amazon S3 object.
     *
     * @param s3          an instance of the {@link S3Client} class used to
     *                    interact with the Amazon S3 service
     * @param bucketName the name of the Amazon S3 bucket where the object is
     *                    stored
     * @param keyName     the name of the Amazon S3 object to be checked
     */
}
```

```

    * @throws S3Exception if an error occurs while interacting with the Amazon
    S3 service
    */
    public static void checkStatus(S3Client s3, String bucketName, String
    keyName) {
        try {
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            HeadObjectResponse response = s3.headObject(headObjectRequest);
            System.out.println("The Amazon S3 object restoration status is " +
            response.restore());

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Per i dettagli sull'API, [HeadObject](#) consulta AWS SDK for Java 2.x API Reference.

Ruby

SDK per Ruby

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectExistsWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An Amazon S3 object.

```

```

def initialize(object)
  @object = object
end

# Checks whether the object exists.
#
# @return [Boolean] True if the object exists; otherwise false.
def exists?
  @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object
#{@object.bucket.name}:#{@object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  exists = wrapper.exists?

  puts "Object #{object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__

```

- Per i dettagli sull'API, [HeadObject](#) consulta AWS SDK per Ruby API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **ListBucketAnalyticsConfigurations** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare ListBucketAnalyticsConfigurations.

CLI

AWS CLI

Come recuperare un elenco di configurazioni di analisi per un bucket

L'esempio `list-bucket-analytics-configurations` seguente recupera un elenco di configurazioni di analisi per il bucket specificato.

```
aws s3api list-bucket-analytics-configurations \
  --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "AnalyticsConfigurationList": [
    {
      "StorageClassAnalysis": {},
      "Id": "1"
    }
  ],
  "IsTruncated": false
}
```

- Per i dettagli sull'API, consulta [ListBucketAnalyticsConfigurations AWS CLI Command Reference](#).

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando restituisce i dettagli delle prime 100 configurazioni di analisi del bucket S3 specificato.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [ListBucketAnalyticsConfigurations](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando restituisce i dettagli delle prime 100 configurazioni di analisi del bucket S3 specificato.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [ListBucketAnalyticsConfigurations](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **ListBucketInventoryConfigurations** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare ListBucketInventoryConfigurations.

CLI

AWS CLI

Come recuperare un elenco di configurazioni di inventario per un bucket

L'esempio list-bucket-inventory-configurations seguente recupera la configurazione dell'inventario per il bucket specificato.

```
aws s3api list-bucket-inventory-configurations \  
  --bucket amzn-s3-demo-bucket
```

Output:

```
{  
  "InventoryConfigurationList": [  
    {  
      "IsEnabled": true,  
      "Destination": {  
        "S3BucketDestination": {  
          "Format": "ORC",  
          "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
```

```

        "AccountId": "123456789012"
      },
      {
        "IncludedObjectVersions": "Current",
        "Id": "1",
        "Schedule": {
          "Frequency": "Weekly"
        }
      },
      {
        "IsEnabled": true,
        "Destination": {
          "S3BucketDestination": {
            "Format": "CSV",
            "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
            "AccountId": "123456789012"
          }
        },
        "IncludedObjectVersions": "Current",
        "Id": "2",
        "Schedule": {
          "Frequency": "Daily"
        }
      }
    ],
    "IsTruncated": false
  }

```

- Per i dettagli sull'API, consulta [ListBucketInventoryConfigurations AWS CLI](#) Command Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando restituisce le prime 100 configurazioni dell'inventario del bucket S3 specificato.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [ListBucketInventoryConfigurations](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando restituisce le prime 100 configurazioni dell'inventario del bucket S3 specificato.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Per i dettagli sull'API, vedere [ListBucketInventoryConfigurations](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **ListBuckets** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare ListBuckets.

.NET

SDK per .NET

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
namespace ListBucketsExample
{
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example uses the AWS SDK for .NET to list the Amazon Simple Storage
    /// Service (Amazon S3) buckets belonging to the default account.
    /// </summary>
```

```

public class ListBuckets
{
    private static IAmazonS3 _s3Client;

    /// <summary>
    /// Get a list of the buckets owned by the default user.
    /// </summary>
    /// <param name="client">An initialized Amazon S3 client object.</param>
    /// <returns>The response from the ListingBuckets call that contains a
    /// list of the buckets owned by the default user.</returns>
    public static async Task<ListBucketsResponse> GetBuckets(IAmazonS3
client)
    {
        return await client.ListBucketsAsync();
    }

    /// <summary>
    /// This method lists the name and creation date for the buckets in
    /// the passed List of S3 buckets.
    /// </summary>
    /// <param name="bucketList">A List of S3 bucket objects.</param>
    public static void DisplayBucketList(List<S3Bucket> bucketList)
    {
        bucketList
            .ForEach(b => Console.WriteLine($"Bucket name: {b.BucketName},
created on: {b.CreationDate}"));
    }

    public static async Task Main()
    {
        // The client uses the AWS Region of the default user.
        // If the Region where the buckets were created is different,
        // pass the Region to the client constructor. For example:
        // _s3Client = new AmazonS3Client(RegionEndpoint.USEast1);
        _s3Client = new AmazonS3Client();
        var response = await GetBuckets(_s3Client);
        DisplayBucketList(response.Buckets);
    }
}
}

```

- Per i dettagli sull'API, [ListBuckets](#) consulta AWS SDK per .NET API Reference.

C++

SDK per C++

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
bool AwsDoc::S3::listBuckets(const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);

    auto outcome = client.ListBuckets();

    bool result = true;
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = false;
    } else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size() << "
buckets\n";
        for (auto &&b: outcome.GetResult().GetBuckets()) {
            std::cout << b.GetName() << std::endl;
        }
    }

    return result;
}
```

- Per i dettagli sull'API, [ListBuckets](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Il comando seguente utilizza il comando `list-buckets` per visualizzare i nomi di tutti i bucket Amazon S3 (in tutte le Regioni).

```
aws s3api list-buckets --query "Buckets[].Name"
```

L'opzione `query` filtra l'output di `list-buckets` restituendo solo nomi dei bucket.

Per ulteriori informazioni sui bucket, consulta “Utilizzo dei bucket Amazon S3” nella Guida per gli sviluppatori di Amazon S3.

- Per i dettagli sull'API, consulta [ListBuckets AWS CLI Command Reference](#).

Go

SDK per Go V2

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.
```

```
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket,
error) {
    var err error
    var output *s3.ListBucketsOutput
    var buckets []types.Bucket
    bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
&s3.ListBucketsInput{})
    for bucketPaginator.HasMorePages() {
        output, err = bucketPaginator.NextPage(ctx)
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
                fmt.Println("You don't have permission to list buckets for this account.")
                err = apiErr
            } else {
                log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
            }
            break
        } else {
            buckets = append(buckets, output.Buckets...)
        }
    }
    return buckets, err
}
```

- Per i dettagli sull'API, [ListBuckets](#) consulta AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListBucketsIterable;
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);
    }

    /**
     * Lists all the S3 buckets available in the current AWS account.
     *
     * @param s3 The {@link S3Client} instance to use for interacting with the
     * Amazon S3 service.
     */
    public static void listAllBuckets(S3Client s3) {
        ListBucketsIterable response = s3.listBucketsPaginator();
        response.buckets().forEach(bucket ->
```

```
        System.out.println("Bucket Name: " + bucket.name());  
    }
```

- Per i dettagli sull'API, [ListBuckets](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elenca i bucket.

```
import {  
    paginateListBuckets,  
    S3Client,  
    S3ServiceException,  
} from "@aws-sdk/client-s3";  
  
/**  
 * List the Amazon S3 buckets in your account.  
 */  
export const main = async () => {  
    const client = new S3Client({});  
    /** @type {?import('@aws-sdk/client-s3').Owner} */  
    let Owner = null;  
  
    /** @type {import('@aws-sdk/client-s3').Bucket[]} */  
    const Buckets = [];  
  
    try {  
        const paginator = paginateListBuckets({ client }, {});  
  
        for await (const page of paginator) {  
            if (!Owner) {  
                Owner = page.Owner;  
            }  
        }  
    }  
}
```

```
Buckets.push(...page.Buckets);
}

console.log(
  `${Owner.DisplayName} owns ${Buckets.length} bucket${
    Buckets.length === 1 ? "" : "s"
  }:`,
);
console.log(`${Buckets.map((b) => ` • ${b.Name}`).join("\n")}`);
} catch (caught) {
  if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while listing buckets.  ${caught.name}:
${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [ListBuckets](#) consulta AWS SDK per JavaScript API Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando restituisce tutti i bucket S3.

```
Get-S3Bucket
```

Esempio 2: questo comando restituisce un bucket denominato “test-files”

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Per i dettagli sull'API, vedere [ListBuckets](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando restituisce tutti i bucket S3.

```
Get-S3Bucket
```

Esempio 2: questo comando restituisce un bucket denominato "test-files"

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Per i dettagli sull'API, vedere [ListBuckets](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Python

SDK per Python (Boto3)

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    @staticmethod
    def list(s3_resource):
        """
        Get the buckets in all Regions for the current account.
```

```

        :param s3_resource: A Boto3 S3 resource. This is a high-level resource in
Boto3
                           that contains collections and factory methods to
create
                           other high-level S3 sub-resources.
        :return: The list of buckets.
        """
        try:
            buckets = list(s3_resource.buckets.all())
            logger.info("Got buckets: %s.", buckets)
        except ClientError:
            logger.exception("Couldn't get buckets.")
            raise
        else:
            return buckets

```

- Per i dettagli sull'API, consulta [ListBuckets AWS SDK for Python \(Boto3\) API Reference](#).

Ruby

SDK per Ruby

Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

require 'aws-sdk-s3'

# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end
end

```



```

# Lists buckets for the current account.
#
# @param count [Integer] The maximum number of buckets to list.
def list_buckets(count)
  puts 'Found these buckets:'
  @s3_resource.buckets.each do |bucket|
    puts "\t#{bucket.name}"
    count -= 1
    break if count.zero?
  end
  true
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list buckets. Here's why: #{e.message}"
  false
end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__

```

- Per i dettagli sull'API, [ListBuckets](#) consulta AWS SDK per Ruby API Reference.

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

async fn show_buckets(
  strict: bool,
  client: &Client,
  region: BucketLocationConstraint,

```

```

) -> Result<(), S3ExampleError> {
    let mut buckets = client.list_buckets().into_paginator().send();

    let mut num_buckets = 0;
    let mut in_region = 0;

    while let Some(Ok(output)) = buckets.next().await {
        for bucket in output.buckets() {
            num_buckets += 1;
            if strict {
                let r = client
                    .get_bucket_location()
                    .bucket(bucket.name().unwrap_or_default())
                    .send()
                    .await?;

                if r.location_constraint() == Some(&region) {
                    println!("{}", bucket.name().unwrap_or_default());
                    in_region += 1;
                }
            } else {
                println!("{}", bucket.name().unwrap_or_default());
            }
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}

```

- Per i dettagli sulle API, consulta la [ListBuckets](#) guida di riferimento all'API AWS SDK for Rust.

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.  
    oo_result = lo_s3->listbuckets(           " oo_result is returned for  
testing purposes. "  
    ).  
    DATA(lv_bucket_count) = lines( oo_result->get_buckets( ) ).  
    MESSAGE |Retrieved { lv_bucket_count } buckets in all regions.| TYPE 'I'.  
CATCH /aws1/cx_rt_generic.  
    MESSAGE 'Unable to list buckets.' TYPE 'E'.  
ENDTRY.
```

- Per i dettagli sulle API, [ListBuckets](#) consulta AWS SDK for SAP ABAP API reference.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import AWSS3  
  
/// Return an array containing information about every available bucket.  
///  
/// - Returns: An array of ``S3ClientTypes.Bucket`` objects describing  
/// each bucket.  
public func getAllBuckets() async throws -> [S3ClientTypes.Bucket] {
```

```
    return try await client.listBuckets(input: ListBucketsInput())
}
```

- Per i dettagli sull'API, consulta la [ListBuckets](#) guida di riferimento all'API AWS SDK for Swift.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **ListMultipartUploads** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare ListMultipartUploads.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Eliminare caricamenti in più parti incompleti](#)

CLI

AWS CLI

Il comando seguente elenca tutti i caricamenti in più parti attivi per un bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api list-multipart-uploads --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "Uploads": [
    {
      "Initiator": {
        "DisplayName": "username",
        "ID": "arn:aws:iam::0123456789012:user/username"
      },
      "Initiated": "2015-06-02T18:01:30.000Z",
      "UploadId":
        "dfRtDYU0WwCCcH43C3WfbkR0NycyCpTJJvxu2i5GYkZ1jF.Yxwh6XG7WfS2vC4to6HiV6Yj1x.cph0gtNBtJ8P3",
      "StorageClass": "STANDARD",
    }
  ]
}
```

```

        "Key": "multipart/01",
        "Owner": {
            "DisplayName": "aws-account-name",
            "ID":
"100719349fc3b6dcd7c820a124bf7aecd408092c3d7b51b38494939801fc248b"
        }
    },
    "CommonPrefixes": []
}

```

I caricamenti in più parti in corso generano costi di archiviazione in Amazon S3. Completa o annulla un caricamento in più parti attivo per rimuoverne le parti dall'account corrente.

- Per i dettagli sull'API, consulta [ListMultipartUploads AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html

```

```
*/

public class ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The name of the Amazon S3 bucket where an in-
progress multipart upload is occurring.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
        listUploads(s3, bucketName);
        s3.close();
    }

    /**
     * Lists the multipart uploads currently in progress in the specified Amazon
S3 bucket.
     *
     * @param s3 the S3Client object used to interact with Amazon S3
     * @param bucketName the name of the Amazon S3 bucket to list the multipart
uploads for
     */
    public static void listUploads(S3Client s3, String bucketName) {
        try {
            ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
                .bucket(bucketName)
                .build();
        }
    }
}
```

```
        ListMultipartUploadsResponse response =
s3.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();
        for (MultipartUpload upload : uploads) {
            System.out.println("Upload in progress: Key = \"" + upload.key()
+ "\", id = " + upload.uploadId());
        }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [ListMultipartUploads](#) consulta AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **ListObjectVersions** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare ListObjectVersions.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- [Guida introduttiva allo storage di oggetti](#)
- [Utilizzo degli oggetti con versione](#)

.NET

SDK per .NET

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example lists the versions of the objects in a version enabled
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class ListObjectVersions
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";

        // If the AWS Region where your bucket is defined is different from
        // the AWS Region where the Amazon S3 bucket is defined, pass the
constant
        // for the AWS Region to the client constructor like this:
        //     var client = new AmazonS3Client(RegionEndpoint.USWest2);
        IAmazonS3 client = new AmazonS3Client();
        await GetObjectListWithAllVersionsAsync(client, bucketName);
    }

    /// <summary>
    /// This method lists all versions of the objects within an Amazon S3
    /// version enabled bucket.
    /// </summary>
    /// <param name="client">The initialized client object used to call
    /// ListVersionsAsync.</param>
    /// <param name="bucketName">The name of the version enabled Amazon S3
bucket
    /// for which you want to list the versions of the contained objects.</
param>
    public static async Task GetObjectListWithAllVersionsAsync(IAmazonS3
client, string bucketName)
    {
        try
        {
            // When you instantiate the ListVersionRequest, you can
            // optionally specify a key name prefix in the request
            // if you want a list of object versions of a specific object.

```



```
// For this example we set a small limit in MaxKeys to return
// a small list of versions.
ListVersionsRequest request = new ListVersionsRequest()
{
    BucketName = bucketName,
    MaxKeys = 2,
};

do
{
    ListVersionsResponse response = await
client.ListVersionsAsync(request);

    // Process response.
    foreach (S3ObjectVersion entry in response.Versions)
    {
        Console.WriteLine($"key: {entry.Key} size:
{entry.Size}");
    }

    // If response is truncated, set the marker to get the next
    // set of keys.
    if (response.IsTruncated)
    {
        request.KeyMarker = response.NextKeyMarker;
        request.VersionIdMarker = response.NextVersionIdMarker;
    }
    else
    {
        request = null;
    }
}
while (request != null);
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error: '{ex.Message}'");
}
}
```

- Per i dettagli sull'API, [ListObjectVersions](#) consulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Il comando seguente recupera le informazioni sulla versione di un oggetto in un bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket --prefix index.html
```

Output:

```
{
  "DeleteMarkers": [
    {
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": true,
      "VersionId": "B2VsEK5saUNNHKc0AJj7hIE86RozToyq",
      "Key": "index.html",
      "LastModified": "2015-11-10T00:57:03.000Z"
    },
    {
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": false,
      "VersionId": ".FLQEZscLIcfxSq.jsFJ.szUkmng2Yw6",
      "Key": "index.html",
      "LastModified": "2015-11-09T23:32:20.000Z"
    }
  ],
  "Versions": [
    {
      "LastModified": "2015-11-10T00:20:11.000Z",
      "VersionId": "Rb_l2T8UHDkFEwCgJjhlGPOZC0qJ.vpD",
      "ETag": "\"0622528de826c0df5db1258a23b80be5\"",
      "StorageClass": "STANDARD",
      "Key": "index.html",
    }
  ]
}
```

```

      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": false,
      "Size": 38
    },
    {
      "LastModified": "2015-11-09T23:26:41.000Z",
      "VersionId": "rasWWGpgk9E4s0LyTJgusGeRQKLVIAff",
      "ETag": "\"06225825b8028de826c0df5db1a23be5\"",
      "StorageClass": "STANDARD",
      "Key": "index.html",
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": false,
      "Size": 38
    },
    {
      "LastModified": "2015-11-09T22:50:50.000Z",
      "VersionId": "null",
      "ETag": "\"d1f45267a863c8392e07d24dd592f1b9\"",
      "StorageClass": "STANDARD",
      "Key": "index.html",
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": false,
      "Size": 533823
    }
  ]
}

```

- Per i dettagli sull'API, consulta [ListObjectVersions AWS CLI](#) Command Reference.

Go

SDK per Go V2

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
    ([]types.ObjectVersion, error) {
    var err error
    var output *s3.ListObjectVersionsOutput
    var versions []types.ObjectVersion
    input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
    versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
```

```
for versionPaginator.HasMorePages() {
    output, err = versionPaginator.NextPage(ctx)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
        break
    } else {
        versions = append(versions, output.Versions...)
    }
}
return versions, err
}
```

- Per i dettagli sull'API, [ListObjectVersions](#) consulta AWS SDK per Go API Reference.

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
async fn show_versions(client: &Client, bucket: &str) -> Result<(), Error> {
    let resp = client.list_object_versions().bucket(bucket).send().await?;

    for version in resp.versions() {
        println!("{}", version.key().unwrap_or_default());
        println!(" version ID: {}", version.version_id().unwrap_or_default());
        println!();
    }

    Ok(())
}
```

- Per i dettagli sulle API, consulta la [ListObjectVersions](#) guida di riferimento all'API AWS SDK for Rust.

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.  
    oo_result = lo_s3->listobjectversions(          " oo_result is returned  
for testing purposes. "  
        iv_bucket = iv_bucket_name  
        iv_prefix = iv_prefix ).  
    MESSAGE 'Retrieved object versions.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Per i dettagli sulle API, [ListObjectVersions](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **ListObjects** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare ListObjects.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Creare una pagina web che elenca gli oggetti Amazon S3](#)

CLI

AWS CLI

L'esempio seguente utilizza il comando `list-objects` per visualizzare i nomi di tutti gli oggetti del bucket specificato:

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size: Size}'
```

L'esempio utilizza l'argomento `--query` per filtrare l'output di `list-objects` fino al valore della chiave e alla dimensione per ogni oggetto

Per ulteriori informazioni sugli oggetti, consulta Operazioni con gli oggetti di Amazon S3 nella Guida per gli sviluppatori di Amazon S3.

- Per i dettagli sull'API, consulta [ListObjects AWS CLI](#) Command Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando recupera le informazioni su tutti gli elementi presenti nel bucket "test-files".

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Esempio 2: questo comando recupera le informazioni sull'elemento "sample.txt" dal bucket "test-files".

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Esempio 3: questo comando recupera le informazioni su tutti gli elementi con il prefisso "sample" dal bucket "test-files".

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Per i dettagli sull'API, vedere [ListObjects](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando recupera le informazioni su tutti gli elementi presenti nel bucket "test-files".

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Esempio 2: questo comando recupera le informazioni sull'elemento "sample.txt" dal bucket "test-files".

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Esempio 3: questo comando recupera le informazioni su tutti gli elementi con il prefisso "sample" dal bucket "test-files".

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Per i dettagli sull'API, vedere [ListObjects](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Usa **ListObjectsV2** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare ListObjectsV2.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- [Informazioni di base](#)
- [Eliminare tutti gli oggetti in un bucket](#)
- [Guida introduttiva allo storage di oggetti](#)

.NET

SDK per .NET (v4)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>
/// Shows how to list the objects in an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket for which to list.
/// <param name="printList">True to print out the list.
/// <returns>The collection of objects.</returns>
public async Task<List<S3Object>?> ListBucketContentsAsync(string bucketName,
bool printList = true)
{
    try
    {
        var request = new ListObjectsV2Request
        {
            BucketName = bucketName,
            MaxKeys = 5,
        };

        if (printList)
        {
            Console.WriteLine("-----");
            Console.WriteLine($"Listing the contents of {bucketName}:");
            Console.WriteLine("-----");
        }

        var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
        {
            BucketName = bucketName,
        });
        var s3Objects = new List<S3Object>();
        await foreach (var response in listObjectsV2Paginator.Responses)
```

```
        {
            if (response.S3Objects != null)
            {
                s3objects.AddRange(response.S3Objects);
            }
        }

        if (printList)
        {
            Console.WriteLine($"Number of Objects: {s3objects.Count}");
            foreach (var entry in s3objects)
            {
                Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
            }
        }

        return s3objects;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error encountered on server.
        Message: '{ex.Message}' getting list of objects.");
        return null;
    }
}
```

- Per i dettagli sull'API, consulta la [ListObjectsversione V2](#) in AWS SDK per .NET API Reference.

SDK per .NET

Note

C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elencare gli oggetti con un impaginatore.

```
using System;
using System.Threading.Tasks;
```

```
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// The following example lists objects in an Amazon Simple Storage
/// Service (Amazon S3) bucket.
/// </summary>
public class ListObjectsPaginator
{
    private const string BucketName = "amzn-s3-demo-bucket";

    public static async Task Main()
    {
        IAmazonS3 s3Client = new AmazonS3Client();

        Console.WriteLine($"Listing the objects contained in {BucketName}:
\n");
        await ListingObjectsAsync(s3Client, BucketName);
    }

    /// <summary>
    /// This method uses a paginator to retrieve the list of objects in an
    /// an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">An Amazon S3 client object.</param>
    /// <param name="bucketName">The name of the S3 bucket whose objects
    /// you want to list.</param>
    public static async Task ListingObjectsAsync(IAmazonS3 client, string
bucketName)
    {
        var listObjectsV2Paginator = client.Paginators.ListObjectsV2(new
ListObjectsV2Request
        {
            BucketName = bucketName,
        });

        await foreach (var response in listObjectsV2Paginator.Responses)
        {
            Console.WriteLine($"HttpStatusCode: {response.HttpStatusCode}");
            Console.WriteLine($"Number of Keys: {response.KeyCount}");
            foreach (var entry in response.S3Objects)
            {
                Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
            }
        }
    }
}
```

```

    }
  }
}

```

- Per i dettagli sull'API, consulta la [ListObjectsversione V2](#) in AWS SDK per .NET API Reference.

Bash

AWS CLI con lo script Bash

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#     $1 - The name of the bucket.
#
# Returns:
#     The list of files in text format.
#
# And:

```

```
#      0 - If successful.
#      1 - If it fails.
#####
function list_items_in_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api list-objects \
        --bucket "$bucket_name" \
        --output text \
        --query 'Contents[].{Key: Key, Size: Size}')

    # shellcheck disable=SC2181
    if [[ ${?} -eq 0 ]]; then
        echo "$response"
    else
        errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
        return 1
    fi
}
```

- Per i dettagli sull'API, vedi [ListObjectsV2](#) in AWS CLI Command Reference.

C++

SDK per C++

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
bool AwsDoc::S3::listObjects(const Aws::String &bucketName,
                             Aws::Vector<Aws::String> &keysResult,
                             const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);
```

```

    Aws::String continuationToken; // Used for pagination.
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }

        auto outcome = s3Client.ListObjectsV2(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: listObjects: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        } else {
            Aws::Vector<Aws::S3::Model::Object> objects =
                outcome.GetResult().GetContents();

            allObjects.insert(allObjects.end(), objects.begin(), objects.end());
            continuationToken = outcome.GetResult().GetNextContinuationToken();
        }
    } while (!continuationToken.empty());

    std::cout << allObjects.size() << " object(s) found:" << std::endl;

    for (const auto &object: allObjects) {
        std::cout << "  " << object.GetKey() << std::endl;
        keysResult.push_back(object.GetKey());
    }

    return true;
}

```

- Per i dettagli sull'API, consulta la [ListObjectsversione V2](#) in AWS SDK per C++ API Reference.

CLI

AWS CLI

Come ottenere un elenco di oggetti in un bucket

L'esempio `list-objects-v2` seguente elenca gli oggetti nel bucket specificato.

```
aws s3api list-objects-v2 \  
  --bucket amzn-s3-demo-bucket
```

Output:

```
{  
  "Contents": [  
    {  
      "LastModified": "2019-11-05T23:11:50.000Z",  
      "ETag": "\"621503c373607d548b37cff8778d992c\"",  
      "StorageClass": "STANDARD",  
      "Key": "doc1.rtf",  
      "Size": 391  
    },  
    {  
      "LastModified": "2019-11-05T23:11:50.000Z",  
      "ETag": "\"a2cecc36ab7c7fe3a71a273b9d45b1b5\"",  
      "StorageClass": "STANDARD",  
      "Key": "doc2.rtf",  
      "Size": 373  
    },  
    {  
      "LastModified": "2019-11-05T23:11:50.000Z",  
      "ETag": "\"08210852f65a2e9cb999972539a64d68\"",  
      "StorageClass": "STANDARD",  
      "Key": "doc3.rtf",  
      "Size": 399  
    },  
    {  
      "LastModified": "2019-11-05T23:11:50.000Z",  
      "ETag": "\"d1852dd683f404306569471af106988e\"",  
      "StorageClass": "STANDARD",  
      "Key": "doc4.rtf",  
      "Size": 6225  
    }  
  ]  
}
```

- Per i dettagli sull'API, consulta [ListObjectsV2](#) in AWS CLI Command Reference.

Go

SDK per Go V2

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform  
// bucket  
// and object actions.  
type BucketBasics struct {  
    S3Client *s3.Client  
}  
  
// ListObjects lists the objects in a bucket.
```



```
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
([]types.Object, error) {
    var err error
    var output *s3.ListObjectsV2Output
    input := &s3.ListObjectsV2Input{
        Bucket: aws.String(bucketName),
    }
    var objects []types.Object
    objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
    for objectPaginator.HasMorePages() {
        output, err = objectPaginator.NextPage(ctx)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucketName)
                err = noBucket
            }
            break
        } else {
            objects = append(objects, output.Contents...)
        }
    }
    return objects, err
}
```

- Per i dettagli sull'API, consulta la [ListObjectsversione V2](#) in AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

/**
 * Asynchronously lists all objects in the specified S3 bucket.
 *
 * @param bucketName the name of the S3 bucket to list objects for
 * @return a {@link CompletableFuture} that completes when all objects have
 * been listed
 */
public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
    ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .maxKeys(1)
        .build();

    ListObjectsV2Publisher paginator =
getAsyncClient().listObjectsV2Paginator(initialRequest);
    return paginator.subscribe(response -> {
        response.contents().forEach(s3Object -> {
            logger.info("Object key: " + s3Object.key());
        });
    }).thenRun(() -> {
        logger.info("Successfully listed all objects in the bucket: " +
bucketName);
    }).exceptionally(ex -> {
        throw new RuntimeException("Failed to list objects", ex);
    });
}

```

Elenca gli oggetti usando l'impaginazione.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
            <bucketName>\s

```

```
        Where:
            bucketName - The Amazon S3 bucket from which objects are read.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    listBucketObjects(s3, bucketName);
    s3.close();
}

/**
 * Lists the objects in the specified S3 bucket.
 *
 * @param s3 the S3Client instance used to interact with Amazon S3
 * @param bucketName the name of the S3 bucket to list the objects from
 */
public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsV2Request listReq = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
        listRes.stream()
            .flatMap(r -> r.contents().stream())
            .forEach(content -> System.out.println(" Key: " + content.key() +
" size = " + content.size()));

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- Per i dettagli sull'API, consulta la [ListObjectsversione V2](#) in AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elenca tutti gli oggetti nel bucket. Se è presente più di un oggetto, `IsTruncated` `NextContinuationToken` verrà utilizzato per scorrere l'elenco completo.

```
import {
  S3Client,
  S3ServiceException,
  // This command supersedes the ListObjectsCommand and is the recommended way to
  list objects.
  paginateListObjectsV2,
} from "@aws-sdk/client-s3";

/**
 * Log all of the object keys in a bucket.
 * @param {{ bucketName: string, pageSize: string }}
 */
export const main = async ({ bucketName, pageSize }) => {
  const client = new S3Client({});
  /** @type {string[][]} */
  const objects = [];
  try {
    const paginator = paginateListObjectsV2(
      { client, /* Max items per page */ pageSize: Number.parseInt(pageSize) },
      { Bucket: bucketName },
    );
```

```
for await (const page of paginator) {
    objects.push(page.Contents.map((o) => o.Key));
}
objects.forEach((objectList, pageNum) => {
    console.log(
        `Page ${pageNum + 1}\n-----\n${objectList.map((o) => `
${o}`)}.join("\n")}\n`,
    );
});
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while listing objects for "${bucketName}". The bucket
doesn't exist.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while listing objects for "${bucketName}".
${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};
```

- Per i dettagli sull'API, consulta [ListObjectsV2](#) in AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

Note

C'è di più su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun listBucketObjects(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The object is ${myObject.size?.let { calKb(it) }} KBs")
            println("The owner is ${myObject.owner}")
        }
    }
}

private fun calKb(intValue: Long): Long = intValue / 1024
```

- Per i dettagli sull'API, consulta [ListObjectsV2](#) in AWS SDK per il riferimento all'API Kotlin.

PHP

SDK per PHP

Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elenca gli oggetti in un bucket.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $contents = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    echo "The contents of your bucket are: \n";
    foreach ($contents['Contents'] as $content) {
```

```

        echo $content['Key'] . "\n";
    }
} catch (Exception $exception) {
    echo "Failed to list objects in $this->bucketName with error: " .
    $exception->getMessage();
    exit("Please fix error with listing objects before continuing.");
}

```

- Per i dettagli sull'API, consulta la [ListObjectsversione V2](#) in AWS SDK per PHP API Reference.

Python

SDK per Python (Boto3)

Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    @staticmethod
    def list(bucket, prefix=None):
        """
        Lists the objects in a bucket, optionally filtered by a prefix.

        :param bucket: The bucket to query. This is a Boto3 Bucket resource.

```

```
        :param prefix: When specified, only objects that start with this prefix
        are listed.
        :return: The list of objects.
        """
        try:
            if not prefix:
                objects = list(bucket.objects.all())
            else:
                objects = list(bucket.objects.filter(Prefix=prefix))
            logger.info(
                "Got objects %s from bucket '%s'", [o.key for o in objects],
                bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get objects for bucket '%s'.",
                bucket.name)
            raise
        else:
            return objects
```

- Per i dettagli sull'API, consulta [ListObjectsV2 nella Guida](#) di riferimento all'API AWS SDK for Python (Boto3).

Ruby

SDK per Ruby

Note

C'è di più su. [GitHub Trova](#) l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
  attr_reader :bucket
```



```

# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
def initialize(bucket)
  @bucket = bucket
end

# Lists object in a bucket.
#
# @param max_objects [Integer] The maximum number of objects to list.
# @return [Integer] The number of objects listed.
def list_objects(max_objects)
  count = 0
  puts "The objects in #{@bucket.name} are:"
  @bucket.objects.each do |obj|
    puts "\t#{obj.key}"
    count += 1
    break if count == max_objects
  end
  count
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list objects in bucket #{bucket.name}. Here's why:
#{e.message}"
  0
end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"

  wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
  count = wrapper.list_objects(25)
  puts "Listed #{count} objects."
end

run_demo if $PROGRAM_NAME == __FILE__

```

- Per i dettagli sull'API, consulta la [ListObjectsversione V2](#) in AWS SDK per Ruby API Reference.

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
pub async fn list_objects(client: &aws_sdk_s3::Client, bucket: &str) ->
    Result<(), S3ExampleError> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!(" - {}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
        }
    }

    Ok(())
}
```

- Per i dettagli sull'API, consulta la [ListObjectsversione 2](#) in AWS SDK for Rust API reference.

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.  
    oo_result = lo_s3->listobjectsv2(           " oo_result is returned for  
testing purposes. "  
    iv_bucket = iv_bucket_name ).  
    MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Per i dettagli sull'API, consulta [ListObjectsV2](#) in AWS SDK per il riferimento all'API SAP ABAP.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import AWSS3  
  
public func listBucketFiles(bucket: String) async throws -> [String] {  
    do {  
        let input = ListObjectsV2Input(  
            bucket: bucket
```

```

    )

    // Use "Paginated" to get all the objects.
    // This lets the SDK handle the 'continuationToken' in
    "ListObjectsV2Output".
    let output = client.listObjectsV2Paginated(input: input)
    var names: [String] = []

    for try await page in output {
        guard let objList = page.contents else {
            print("ERROR: listObjectsV2Paginated returned nil contents.")
            continue
        }

        for obj in objList {
            if let objName = obj.key {
                names.append(objName)
            }
        }
    }

    return names
}
catch {
    print("ERROR: ", dump(error, name: "Listing objects."))
    throw error
}
}

```

- Per i dettagli sull'API, consulta la [ListObjectsversione 2](#) in AWS SDK for Swift API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutBucketAccelerateConfiguration** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare PutBucketAccelerateConfiguration.

.NET

SDK per .NET

Note

C'è altro su GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// Amazon Simple Storage Service (Amazon S3) Transfer Acceleration is a
/// bucket-level feature that enables you to perform faster data transfers
/// to Amazon S3. This example shows how to configure Transfer
/// Acceleration.
/// </summary>
public class TransferAcceleration
{
    /// <summary>
    /// The main method initializes the client object and sets the
    /// Amazon Simple Storage Service (Amazon S3) bucket name before
    /// calling EnableAccelerationAsync.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        const string bucketName = "amzn-s3-demo-bucket";

        await EnableAccelerationAsync(s3Client, bucketName);
    }

    /// <summary>
    /// This method sets the configuration to enable transfer acceleration
    /// for the bucket referred to in the bucketName parameter.
    /// </summary>
    /// <param name="client">An Amazon S3 client used to enable the
    /// acceleration on an Amazon S3 bucket.</param>
```

```

    the
    /// <param name="bucketName">The name of the Amazon S3 bucket for which
    /// method will be enabling acceleration.</param>
    private static async Task EnableAccelerationAsync(AmazonS3Client client,
    string bucketName)
    {
        try
        {
            var putRequest = new PutBucketAccelerateConfigurationRequest
            {
                BucketName = bucketName,
                AccelerateConfiguration = new AccelerateConfiguration
                {
                    Status = BucketAccelerateStatus.Enabled,
                },
            };
            await client.PutBucketAccelerateConfigurationAsync(putRequest);

            var getRequest = new GetBucketAccelerateConfigurationRequest
            {
                BucketName = bucketName,
            };
            var response = await
client.GetBucketAccelerateConfigurationAsync(getRequest);

            Console.WriteLine($"Acceleration state = '{response.Status}' ");
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error occurred. Message: '{ex.Message}' when
setting transfer acceleration");
        }
    }
}

```

- Per i dettagli sull'API, [PutBucketAccelerateConfiguration](#) consulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Come impostare la configurazione di accelerazione di un bucket

L'esempio `put-bucket-accelerate-configuration` seguente abilita la configurazione di accelerazione per il bucket specificato.

```
aws s3api put-bucket-accelerate-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --accelerate-configuration Status=Enabled
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [PutBucketAccelerateConfiguration AWS CLI Command Reference](#).

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando abilita l'accelerazione del trasferimento per il bucket S3 specificato.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- Per i dettagli sull'API, vedere [PutBucketAccelerateConfiguration](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando abilita l'accelerazione del trasferimento per il bucket S3 specificato.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- Per i dettagli sull'API, vedere [PutBucketAccelerateConfiguration](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutBucketAcl** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare PutBucketAcl.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Gestire le liste di controllo degli accessi \(ACL\)](#)

.NET

SDK per .NET

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>
/// Creates an Amazon S3 bucket with an ACL to control access to the
/// bucket and the objects stored in it.
/// </summary>
/// <param name="client">The initialized client object used to create
/// an Amazon S3 bucket, with an ACL applied to the bucket.
/// </param>
/// <param name="region">The AWS Region where the bucket will be
created.</param>
/// <param name="newBucketName">The name of the bucket to create.</param>
/// <returns>A boolean value indicating success or failure.</returns>
public static async Task<bool> CreateBucketUseCannedACLAsync(IAmazonS3
client, S3Region region, string newBucketName)
```



```
{
    try
    {
        // Create a new Amazon S3 bucket with Canned ACL.
        var putBucketRequest = new PutBucketRequest()
        {
            BucketName = newBucketName,
            BucketRegion = region,
            CannedACL = S3CannedACL.LogDeliveryWrite,
        };

        PutBucketResponse putBucketResponse = await
client.PutBucketAsync(putBucketRequest);

        return putBucketResponse.HttpStatusCode ==
System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Amazon S3 error: {ex.Message}");
    }

    return false;
}
```

- Per i dettagli sull'API, [PutBucketAcl](#) consulta AWS SDK per .NET API Reference.

C++

SDK per C++

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
bool AwsDoc::S3::putBucketAcl(const Aws::String &bucketName, const Aws::String
&ownerID,
                                const Aws::String &granteePermission,
```

```
const Aws::String &granteeType, const Aws::String
&granteeID,
const Aws::String &granteeEmailAddress,
const Aws::String &granteeURI, const
Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutBucketAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);

    Aws::S3::Model::PutBucketAclOutcome outcome =
        s3Client.PutBucketAcl(request);

    if (!outcome.IsSuccess()) {
```

```

        const Aws::S3::S3Error &error = outcome.GetError();

        std::cerr << "Error: putBucketAcl: " << error.GetExceptionName()
                  << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the bucket '" << bucketName
                  << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
enumeration.
/*!
 \param access: Human readable string.
 \return Permission: A Permission enum.
 */

Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration
 */

Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")

```

```
    return Aws::S3::Model::Type::CanonicalUser;  
    if (type == "Group")  
        return Aws::S3::Model::Type::Group;  
    return Aws::S3::Model::Type::NOT_SET;  
}
```

- Per i dettagli sull'API, [PutBucketAcl](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Questo esempio concede `full control` a due AWS utenti (`user1@example.com` e `user2@example.com`) `read` l'autorizzazione a tutti:

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --grant-full-  
control emailaddress=user1@example.com,emailaddress=user2@example.com --grant-  
read uri=http://acs.amazonaws.com/groups/global/AllUsers
```

Vedi <http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html> i dettagli sugli ACL personalizzati (i comandi ACL `s3api`, ad esempio `put-bucket-acl`, usano la stessa notazione abbreviata degli argomenti).

- Per i dettagli sull'API, consulta Command Reference. [PutBucketAcl](#) AWS CLI

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
```

```
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;

import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetAcl {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
            <bucketName> <id>\s

            Where:
            bucketName - The Amazon S3 bucket to grant permissions on.\s
            id - The ID of the owner of this bucket (you can get this value
from the AWS Management Console).
            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        String bucketName = args[0];
        String id = args[1];
        System.out.format("Setting access \n");
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();
```

```
        setBucketAcl(s3, bucketName, id);
        System.out.println("Done!");
        s3.close();
    }

    /**
     * Sets the Access Control List (ACL) for an Amazon S3 bucket.
     *
     * @param s3 the S3Client instance to be used for the operation
     * @param bucketName the name of the S3 bucket to set the ACL for
     * @param id the ID of the AWS user or account that will be granted full
control of the bucket
     * @throws S3Exception if an error occurs while setting the bucket ACL
     */
    public static void setBucketAcl(S3Client s3, String bucketName, String id) {
        try {
            Grant ownerGrant = Grant.builder()
                .grantee(builder -> builder.id(id)
                    .type(Type.CANONICAL_USER))
                .permission(Permission.FULL_CONTROL)
                .build();

            List<Grant> grantList2 = new ArrayList<>();
            grantList2.add(ownerGrant);

            AccessControlPolicy acl = AccessControlPolicy.builder()
                .owner(builder -> builder.id(id))
                .grants(grantList2)
                .build();

            PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
                .bucket(bucketName)
                .accessControlPolicy(acl)
                .build();

            s3.putBucketAcl(putAclReq);

        } catch (S3Exception e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

- Per i dettagli sull'API, [PutBucketAcl](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Inserisci l'ACL del bucket.

```
import {
  PutBucketAclCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Grant read access to a user using their canonical AWS account ID.
 *
 * Most Amazon S3 use cases don't require the use of access control lists (ACLs).
 * We recommend that you disable ACLs, except in unusual circumstances where
 * you need to control access for each object individually. Consider a policy
 * instead.
 * For more information see https://docs.aws.amazon.com/AmazonS3/latest/
 * userguide/bucket-policies.html.
 * @param {{ bucketName: string, granteeCanonicalUserId: string,
 *   ownerCanonicalUserId }}
 */
export const main = async ({
  bucketName,
  granteeCanonicalUserId,
  ownerCanonicalUserId,
}) => {
  const client = new S3Client({});
  const command = new PutBucketAclCommand({
    Bucket: bucketName,
```

```

    AccessControlPolicy: {
      Grants: [
        {
          Grantee: {
            // The canonical ID of the user. This ID is an obfuscated form of
            // your AWS account number.
            // It's unique to Amazon S3 and can't be found elsewhere.
            // For more information, see https://docs.aws.amazon.com/AmazonS3/latest/userguide/finding-canonical-user-id.html.
            ID: granteeCanonicalUserId,
            Type: "CanonicalUser",
          },
          // One of FULL_CONTROL | READ | WRITE | READ_ACP | WRITE_ACP
          // https://docs.aws.amazon.com/AmazonS3/latest/API/API\_Grant.html#AmazonS3-Type-Grant-Permission
          Permission: "READ",
        },
      ],
      Owner: {
        ID: ownerCanonicalUserId,
      },
    },
  });

  try {
    await client.send(command);
    console.log(`Granted READ access to ${bucketName}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while setting ACL for bucket ${bucketName}. The bucket
        doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while setting ACL for bucket ${bucketName}.
        ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
}

```



```
}  
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [PutBucketAcl](#) consulta AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun setBucketAcl(  
    bucketName: String,  
    idVal: String,  
) {  
    val myGrant =  
        Grantee {  
            id = idVal  
            type = Type.CanonicalUser  
        }  
  
    val ownerGrant =  
        Grant {  
            grantee = myGrant  
            permission = Permission.FullControl  
        }  
  
    val grantList = mutableListOf<Grant>()  
    grantList.add(ownerGrant)  
  
    val ownerOb =  
        Owner {  
            id = idVal  
        }  
}
```

```

val acl =
    AccessControlPolicy {
        owner = ownerObj
        grants = grantList
    }

val request =
    PutBucketAclRequest {
        bucket = bucketName
        accessControlPolicy = acl
    }

S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
    s3.putBucketAcl(request)
    println("An ACL was successfully set on $bucketName")
}
}

```

- Per i dettagli sull'API, [PutBucketAcl](#) consulta AWS SDK for Kotlin API reference.

Python

SDK per Python (Boto3)

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

```

```
def grant_log_delivery_access(self):
    """
    Grant the AWS Log Delivery group write access to the bucket so that
    Amazon S3 can deliver access logs to the bucket. This is the only
recommended
    use of an S3 bucket ACL.
    """
    try:
        acl = self.bucket.Acl()
        # Putting an ACL overwrites the existing ACL. If you want to preserve
        # existing grants, append new grants to the list of existing grants.
        grants = acl.grants if acl.grants else []
        grants.append(
            {
                "Grantee": {
                    "Type": "Group",
                    "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery",
                },
                "Permission": "WRITE",
            }
        )
        acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
        logger.info("Granted log delivery access to bucket '%s'",
self.bucket.name)
    except ClientError:
        logger.exception("Couldn't add ACL to bucket '%s'.",
self.bucket.name)
        raise
```

- Per i dettagli sull'API, consulta [PutBucketAcl AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.  
  " Example: Grant log delivery access to a bucket  
  " iv_grantwrite = 'uri=http://acs.amazonaws.com/groups/s3/LogDelivery'  
  lo_s3->putbucketacl(  
    iv_bucket = iv_bucket_name  
    iv_grantwrite = iv_grantwrite ).  
  MESSAGE 'Bucket ACL updated.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
  MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Per i dettagli sulle API, [PutBucketAcl](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutBucketCors** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare PutBucketCors.

.NET

SDK per .NET

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

    /// <summary>
    /// Add CORS configuration to the Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used
    /// to apply the CORS configuration to an Amazon S3 bucket.</param>
    /// <param name="configuration">The CORS configuration to apply.</param>
    private static async Task PutCORSConfigurationAsync(AmazonS3Client
client, CORSConfiguration configuration)
    {
        PutCORSConfigurationRequest request = new
PutCORSConfigurationRequest()
        {
            BucketName = BucketName,
            Configuration = configuration,
        };

        _ = await client.PutCORSConfigurationAsync(request);
    }

```

- Per i dettagli sull'API, [PutBucketCors](#) consulta AWS SDK per .NET API Reference.

CLI

AWS CLI

L'esempio seguente abilita le richieste PUT, POST e DELETE da `www.example.com` e abilita le richieste GET da qualsiasi dominio:

```
aws s3api put-bucket-cors --bucket amzn-s3-demo-bucket --cors-configuration file://cors.json
```

cors.json:

```

{
  "CORSRules": [
    {
      "AllowedOrigins": ["http://www.example.com"],
      "AllowedHeaders": ["*"],
      "AllowedMethods": ["PUT", "POST", "DELETE"],
      "MaxAgeSeconds": 3000,
    }
  ]
}

```

```

    "ExposeHeaders": ["x-amz-server-side-encryption"]
  },
  {
    "AllowedOrigins": ["*"],
    "AllowedHeaders": ["Authorization"],
    "AllowedMethods": ["GET"],
    "MaxAgeSeconds": 3000
  }
]
}

```

- Per i dettagli sull'API, consulta [PutBucketCors AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;

import java.util.ArrayList;
import java.util.List;

import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>

```

```

* For more information, see the following documentation topic:
* <p>
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class S3Cors {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <accountId>\s

            Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                accountId - The id of the account that owns the Amazon S3 bucket.
            "";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setCorsInformation(s3, bucketName, accountId);
        getBucketCorsInformation(s3, bucketName, accountId);
        deleteBucketCorsInformation(s3, bucketName, accountId);
        s3.close();
    }

    /**
     * Deletes the CORS (Cross-Origin Resource Sharing) configuration for an
     * Amazon S3 bucket.
     *
     * @param s3          the {@link S3Client} instance used to interact with
     *                    the Amazon S3 service
     * @param bucketName  the name of the Amazon S3 bucket for which the CORS
     *                    configuration should be deleted
     * @param accountId   the expected AWS account ID of the bucket owner

```

```
    *
    * @throws S3Exception if an error occurs while deleting the CORS
    configuration for the bucket
    */
    public static void deleteBucketCorsInformation(S3Client s3, String
    bucketName, String accountId) {
        try {
            DeleteBucketCorsRequest bucketCorsRequest =
    DeleteBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            s3.deleteBucketCors(bucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Retrieves the CORS (Cross-Origin Resource Sharing) configuration for the
    specified S3 bucket.
     *
     * @param s3 the S3Client instance to use for the operation
     * @param bucketName the name of the S3 bucket to retrieve the CORS
    configuration for
     * @param accountId the expected bucket owner's account ID
     *
     * @throws S3Exception if there is an error retrieving the CORS configuration
     */
    public static void getBucketCorsInformation(S3Client s3, String bucketName,
    String accountId) {
        try {
            GetBucketCorsRequest bucketCorsRequest =
    GetBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            GetBucketCorsResponse corsResponse =
    s3.getBucketCors(bucketCorsRequest);
            List<CORSRule> corsRules = corsResponse.corsRules();
        }
```



```
        for (CORSRule rule : corsRules) {
            System.out.println("allowOrigins: " + rule.allowedOrigins());
            System.out.println("AllowedMethod: " + rule.allowedMethods());
        }

    } catch (S3Exception e) {

        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

/**
 * Sets the Cross-Origin Resource Sharing (CORS) rules for an Amazon S3
 * bucket.
 *
 * @param s3 The S3Client object used to interact with the Amazon S3 service.
 * @param bucketName The name of the S3 bucket to set the CORS rules for.
 * @param accountId The AWS account ID of the bucket owner.
 */
public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
    List<String> allowMethods = new ArrayList<>();
    allowMethods.add("PUT");
    allowMethods.add("POST");
    allowMethods.add("DELETE");

    List<String> allowOrigins = new ArrayList<>();
    allowOrigins.add("http://example.com");
    try {
        // Define CORS rules.
        CORSRule corsRule = CORSRule.builder()
            .allowedMethods(allowMethods)
            .allowedOrigins(allowOrigins)
            .build();

        List<CORSRule> corsRules = new ArrayList<>();
        corsRules.add(corsRule);
        CORSConfiguration configuration = CORSConfiguration.builder()
            .corsRules(corsRules)
            .build();

        PutBucketCorsRequest putBucketCorsRequest =
PutBucketCorsRequest.builder()
```

```
        .bucket(bucketName)
        .corsConfiguration(configuration)
        .expectedBucketOwner(accountId)
        .build();

    s3.putBucketCors(putBucketCorsRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, [PutBucketCors](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Aggiungi una regola CORS.

```
import {
    PutBucketCorsCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Allows cross-origin requests to an S3 bucket by setting the CORS
 * configuration.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
    const client = new S3Client({});
```

```
try {
  await client.send(
    new PutBucketCorsCommand({
      Bucket: bucketName,
      CORSConfiguration: {
        CORSRules: [
          {
            // Allow all headers to be sent to this bucket.
            AllowedHeaders: ["*"],
            // Allow only GET and PUT methods to be sent to this bucket.
            AllowedMethods: ["GET", "PUT"],
            // Allow only requests from the specified origin.
            AllowedOrigins: ["https://www.example.com"],
            // Allow the entity tag (ETag) header to be returned in the
            response. The ETag header
            // The entity tag represents a specific version of the object. The
            ETag reflects
            // changes only to the contents of an object, not its metadata.
            ExposeHeaders: ["ETag"],
            // How long the requesting browser should cache the preflight
            response. After
            // this time, the preflight request will have to be made again.
            MaxAgeSeconds: 3600,
          },
        ],
      },
    ));
  console.log(`Successfully set CORS rules for bucket: ${bucketName}`);
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "NoSuchBucket"
  ) {
    console.error(
      `Error from S3 while setting CORS rules for ${bucketName}. The bucket
      doesn't exist.`
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while setting CORS rules for ${bucketName}.
      ${caught.name}: ${caught.message}`
    );
  }
}
```

```
    } else {  
        throw caught;  
    }  
}  
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [PutBucketCors](#) consulta AWS SDK per JavaScript API Reference.

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""  
  
    def __init__(self, bucket):  
        """  
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in  
Boto3  
                        that wraps bucket actions in a class-like structure.  
        """  
        self.bucket = bucket  
        self.name = bucket.name  
  
    def put_cors(self, cors_rules):  
        """  
        Apply CORS rules to the bucket. CORS rules specify the HTTP actions that  
are  
        allowed from other domains.  
  
        :param cors_rules: The CORS rules to apply.  
        """
```

```
    try:
        self.bucket.Cors().put(CORSConfiguration={"CORSRules": cors_rules})
        logger.info(
            "Put CORS rules %s for bucket '%s'.", cors_rules,
            self.bucket.name
        )
    except ClientError:
        logger.exception("Couldn't put CORS rules for bucket %s.",
            self.bucket.name)
        raise
```

- Per i dettagli sull'API, consulta [PutBucketCors AWS SDK for Python \(Boto3\) API Reference](#).

Ruby

SDK per Ruby

Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  # an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Sets CORS rules on a bucket.
  #
  # @param allowed_methods [Array<String>] The types of HTTP requests to allow.
  # @param allowed_origins [Array<String>] The origins to allow.
  # @returns [Boolean] True if the CORS rules were set; otherwise, false.
```

```

def set_cors(allowed_methods, allowed_origins)
  @bucket_cors.put(
    cors_configuration: {
      cors_rules: [
        {
          allowed_methods: allowed_methods,
          allowed_origins: allowed_origins,
          allowed_headers: %w[*],
          max_age_seconds: 3600
        }
      ]
    }
  )
  true
rescue Aws::Errors::ServiceError => e
  puts "Couldn't set CORS rules for #{@bucket_cors.bucket.name}. Here's why:
#{e.message}"
  false
end

end

```

- Per i dettagli sull'API, [PutBucketCors](#) consulta AWS SDK per Ruby API Reference.

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

TRY.

```

" Example: Allow PUT, POST, DELETE methods from http://www.example.com
lo_s3->putbucketcors(
  iv_bucket = iv_bucket_name
  io_corsconfiguration = NEW /aws1/cl_s3_corsconfiguration(
    it_corsrules = it_cors_rules ) ).
MESSAGE 'Bucket CORS configuration set.' TYPE 'I'.

```

```
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Per i dettagli sulle API, [PutBucketCors](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutBucketEncryption** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare PutBucketEncryption.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Guida introduttiva allo storage di oggetti](#)

.NET

SDK per .NET

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>  
/// Set the bucket server side encryption to use AWSKMS with a customer-  
managed key id.  
/// </summary>  
/// <param name="bucketName">Name of the bucket.</param>  
/// <param name="kmsKeyId">The Id of the KMS Key.</param>  
/// <returns>True if successful.</returns>  
public static async Task<bool> SetBucketServerSideEncryption(string  
    bucketName, string kmsKeyId)  
{
```

```

        var serverSideEncryptionByDefault = new ServerSideEncryptionConfiguration
        {
            ServerSideEncryptionRules = new List<ServerSideEncryptionRule>
            {
                new ServerSideEncryptionRule
                {
                    ServerSideEncryptionByDefault = new
ServerSideEncryptionByDefault
                    {
                        ServerSideEncryptionAlgorithm =
ServerSideEncryptionMethod.AWSKMS,
                        ServerSideEncryptionKeyManagementServiceKeyId = kmsKeyId
                    }
                }
            };
        try
        {
            var encryptionResponse = await _s3Client.PutBucketEncryptionAsync(new
PutBucketEncryptionRequest
            {
                BucketName = bucketName,
                ServerSideEncryptionConfiguration =
serverSideEncryptionByDefault,
            });

            return encryptionResponse.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine(ex.ErrorCode == "AccessDenied"
                ? $"This account does not have permission to set encryption on
{bucketName}, please try again."
                : $"Unable to set bucket encryption for bucket {bucketName},
{ex.Message}");
        }
        return false;
    }
}

```

- Per i dettagli sull'API, [PutBucketEncryption](#) consulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Come configurare la crittografia lato server per un bucket

L'esempio `put-bucket-encryption` seguente imposta la crittografia AES256 come predefinita per il bucket specificato.

```
aws s3api put-bucket-encryption \  
    --bucket amzn-s3-demo-bucket \  
    --server-side-encryption-configuration '{"Rules":  
    [{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}}]'
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [PutBucketEncryption AWS CLI Command Reference](#).

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando abilita la crittografia AES256 lato server predefinita con Amazon S3 Managed Keys SSE-S3 () sul bucket specificato.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
    @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Per i dettagli sull'API, vedere [PutBucketEncryption](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando abilita la crittografia AES256 lato server predefinita con Amazon S3 Managed Keys SSE-S3 () sul bucket specificato.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
    @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Per i dettagli sull'API, vedere [PutBucketEncryption](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutBucketLifecycleConfiguration** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare PutBucketLifecycleConfiguration.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- [Eliminare caricamenti in più parti incompleti](#)
- [Gestire messaggi di grandi dimensioni con S3](#)
- [Utilizzo degli oggetti con versione](#)

.NET

SDK per .NET

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>
/// Adds lifecycle configuration information to the S3 bucket named in
/// the bucketName parameter.
/// </summary>
/// <param name="client">The S3 client used to call the
/// PutLifecycleConfigurationAsync method.</param>
/// <param name="bucketName">A string representing the S3 bucket to
/// which configuration information will be added.</param>
/// <param name="configuration">A LifecycleConfiguration object that
/// will be applied to the S3 bucket.</param>
```

```
public static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
string bucketName, LifecycleConfiguration configuration)
{
    var request = new PutLifecycleConfigurationRequest()
    {
        BucketName = bucketName,
        Configuration = configuration,
    };
    var response = await client.PutLifecycleConfigurationAsync(request);
}
```

- Per i dettagli sull'API, [PutBucketLifecycleConfiguration](#) consulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Il comando seguente applica una configurazione del ciclo di vita da un bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api put-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --
lifecycle-configuration file://lifecycle.json
```

Il file `lifecycle.json` è un documento JSON nella cartella corrente che specifica due regole:

```
{
  "Rules": [
    {
      "ID": "Move rotated logs to Glacier",
      "Prefix": "rotated/",
      "Status": "Enabled",
      "Transitions": [
        {
          "Date": "2015-11-10T00:00:00.000Z",
          "StorageClass": "GLACIER"
        }
      ]
    }
  ],
}
```

```
{
  "Status": "Enabled",
  "Prefix": "",
  "NoncurrentVersionTransitions": [
    {
      "NoncurrentDays": 2,
      "StorageClass": "GLACIER"
    }
  ],
  "ID": "Move old versions to Glacier"
}
```

La prima regola sposta i file con il prefisso `rotated` su Glacier nella data specificata. La seconda regola sposta le vecchie versioni degli oggetti su Glacier quando non sono più attuali. Per informazioni sui formati di timestamp accettabili, consulta “Specifica dei valori di parametro” nella Guida per l’utente dell’Interfaccia della linea di comando AWS (AWS CLI).

- Per i dettagli sull’API, consulta [PutBucketLifecycleConfiguration AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C’è altro su GitHub. Trova l’esempio completo e scopri di più sulla configurazione e l’esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
  software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
  software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
```

```
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class LifecycleConfiguration {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <accountId>\s

            Where:
                bucketName - The Amazon Simple Storage Service (Amazon S3) bucket
to upload an object into.
                accountId - The id of the account that owns the Amazon S3 bucket.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
```

```
        .build();

        setLifecycleConfig(s3, bucketName, accountId);
        getLifecycleConfig(s3, bucketName, accountId);
        deleteLifecycleConfig(s3, bucketName, accountId);
        System.out.println("You have successfully created, updated, and deleted a
Lifecycle configuration");
        s3.close();
    }

    /**
     * Sets the lifecycle configuration for an Amazon S3 bucket.
     *
     * @param s3          The Amazon S3 client to use for the operation.
     * @param bucketName  The name of the Amazon S3 bucket.
     * @param accountId   The expected owner of the Amazon S3 bucket.
     *
     * @throws S3Exception if there is an error setting the lifecycle
     configuration.
     */
    public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            // Create a rule to archive objects with the "glacierobjects/" prefix
to the

            // S3 Glacier Flexible Retrieval storage class immediately.
            LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
                .prefix("glacierobjects/")
                .build();

            Transition transition = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

            LifecycleRule rule1 = LifecycleRule.builder()
                .id("Archive immediately rule")
                .filter(ruleFilter)
                .transitions(transition)
                .status(ExpirationStatus.ENABLED)
                .build();

            // Create a second rule.
            Transition transition2 = Transition.builder()
```

```
        .storageClass(TransitionStorageClass.GLACIER)
        .days(0)
        .build();

List<Transition> transitionList = new ArrayList<>();
transitionList.add(transition2);

LifecycleRuleFilter ruleFilter2 = LifecycleRuleFilter.builder()
    .prefix("glacierobjects/")
    .build();

LifecycleRule rule2 = LifecycleRule.builder()
    .id("Archive and then delete rule")
    .filter(ruleFilter2)
    .transitions(transitionList)
    .status(ExpirationStatus.ENABLED)
    .build();

// Add the LifecycleRule objects to an ArrayList.
ArrayList<LifecycleRule> ruleList = new ArrayList<>();
ruleList.add(rule1);
ruleList.add(rule2);

BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
    .rules(ruleList)
    .build();

PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
    .builder()
    .bucket(bucketName)
    .lifecycleConfiguration(lifecycleConfiguration)
    .expectedBucketOwner(accountId)
    .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
/**
 * Retrieves the lifecycle configuration for an Amazon S3 bucket and adds a
new lifecycle rule to it.
 *
 * @param s3 the S3Client instance used to interact with Amazon S3
 * @param bucketName the name of the Amazon S3 bucket
 * @param accountId the expected owner of the Amazon S3 bucket
 */
public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
    try {
        GetBucketLifecycleConfigurationRequest
getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
        .builder()
        .bucket(bucketName)
        .expectedBucketOwner(accountId)
        .build();

        GetBucketLifecycleConfigurationResponse response = s3

.getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
        List<LifecycleRule> newList = new ArrayList<>();
        List<LifecycleRule> rules = response.rules();
        for (LifecycleRule rule : rules) {
            newList.add(rule);
        }

        // Add a new rule with both a prefix predicate and a tag predicate.
        LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
            .prefix("YearlyDocuments/")
            .build();

        Transition transition = Transition.builder()
            .storageClass(TransitionStorageClass.GLACIER)
            .days(3650)
            .build();

        LifecycleRule rule1 = LifecycleRule.builder()
            .id("NewRule")
            .filter(ruleFilter)
            .transitions(transition)
            .status(ExpirationStatus.ENABLED)
            .build();
    }
}
```



```

        // Add the new rule to the list.
        newList.add(rule1);
        BucketLifecycleConfiguration lifecycleConfiguration =
BucketLifecycleConfiguration.builder()
            .rules(newList)
            .build();

        PutBucketLifecycleConfigurationRequest
putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
            .builder()
            .bucket(bucketName)
            .lifecycleConfiguration(lifecycleConfiguration)
            .expectedBucketOwner(accountId)
            .build();

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

/**
 * Deletes the lifecycle configuration for an Amazon S3 bucket.
 *
 * @param s3 the {@link S3Client} to use for the operation
 * @param bucketName the name of the S3 bucket
 * @param accountId the expected account owner of the S3 bucket
 *
 * @throws S3Exception if an error occurs while deleting the lifecycle
configuration
 */
public static void deleteLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
    try {
        DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
DeleteBucketLifecycleRequest
            .builder()
            .bucket(bucketName)
            .expectedBucketOwner(accountId)
            .build();

```

```

        s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Per i dettagli sull'API, [PutBucketLifecycleConfiguration](#) consulta AWS SDK for Java 2.x API Reference.

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def put_lifecycle_configuration(self, lifecycle_rules):
        """
        Apply a lifecycle configuration to the bucket. The lifecycle
        configuration can

```

be used to archive or delete the objects in the bucket according to specified

parameters, such as a number of days.

```
:param lifecycle_rules: The lifecycle rules to apply.
"""
```

```
try:
    self.bucket.LifecycleConfiguration().put(
        LifecycleConfiguration={"Rules": lifecycle_rules}
    )
    logger.info(
        "Put lifecycle rules %s for bucket '%s'.",
        lifecycle_rules,
        self.bucket.name,
    )
except ClientError:
    logger.exception(
        "Couldn't put lifecycle rules for bucket '%s'.", self.bucket.name
    )
    raise
```

- Per i dettagli sull'API, consulta [PutBucketLifecycleConfiguration AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

TRY.

```
" Example: Expire objects with prefix 'logs/' after 30 days
lo_s3->putbucketlifecycleconf(
    iv_bucket = iv_bucket_name
    io_lifecycleconfiguration = NEW /aws1/cl_s3_bucketlcconf(
```

```
        it_rules = it_lifecycle_rule ) ).  
        MESSAGE 'Bucket lifecycle configuration set.' TYPE 'I'.  
    CATCH /aws1/cx_s3_nosuchbucket.  
        MESSAGE 'Bucket does not exist.' TYPE 'E'.  
    ENDTRY.
```

- Per i dettagli sulle API, [PutBucketLifecycleConfiguration](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutBucketLogging** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare PutBucketLogging.

.NET

SDK per .NET

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
using System;  
using System.IO;  
using System.Threading.Tasks;  
using Amazon.S3;  
using Amazon.S3.Model;  
using Microsoft.Extensions.Configuration;  
  
/// <summary>  
/// This example shows how to enable logging on an Amazon Simple Storage  
/// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for  
/// this example. The first is the bucket for which you wish to enable  
/// logging, and the second is the location where you want to store the  
/// logs.
```

```
/// </summary>
public class ServerAccessLogging
{
    private static IConfiguration _configuration = null!;

    public static async Task Main()
    {
        LoadConfig();

        string bucketName = _configuration["BucketName"];
        string logBucketName = _configuration["LogBucketName"];
        string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
        string accountId = _configuration["AccountId"];

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
        IAmazonS3 client = new AmazonS3Client();

        try
        {
            // Update bucket policy for target bucket to allow delivery of
logs to it.
            await SetBucketPolicyToAllowLogDelivery(
                client,
                bucketName,
                logBucketName,
                logObjectKeyPrefix,
                accountId);

            // Enable logging on the source bucket.
            await EnableLoggingAsync(
                client,
                bucketName,
                logBucketName,
                logObjectKeyPrefix);
        }
        catch (AmazonS3Exception e)
        {
            Console.WriteLine($"Error: {e.Message}");
        }
    }
}
```

```

    /// <summary>
    /// This method grants appropriate permissions for logging to the
    /// Amazon S3 bucket where the logs will be stored.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client which will be
used
    /// to apply the bucket policy.</param>
    /// <param name="sourceBucketName">The name of the source bucket.</param>
    /// <param name="logBucketName">The name of the bucket where logging
    /// information will be stored.</param>
    /// <param name="logPrefix">The logging prefix where the logs should be
delivered.</param>
    /// <param name="accountId">The account id of the account where the
source bucket exists.</param>
    /// <returns>Async task.</returns>
    public static async Task SetBucketPolicyToAllowLogDelivery(
        IAmazonS3 client,
        string sourceBucketName,
        string logBucketName,
        string logPrefix,
        string accountId)
    {
        var resourceArn = @""arn:aws:s3::" + logBucketName + "/" +
logPrefix + @"""";

        var newPolicy = @"{
            ""Statement"": [{
                ""Sid"": ""S3ServerAccessLogsPolicy"",
                ""Effect"": ""Allow"",
                ""Principal"": { ""Service"":
""logging.s3.amazonaws.com"" },
                ""Action"": [""s3:PutObject""],
                ""Resource"": ["" + resourceArn + @""],
                ""Condition"": {
                    ""ArnLike"": { ""aws:SourceArn"":
""arn:aws:s3::" + sourceBucketName + @"" },
                    ""StringEquals"": { ""aws:SourceAccount"": """" +
accountId + @"""" }
                }
            }]
        }";

        Console.WriteLine($"The policy to apply to bucket {logBucketName} to
enable logging:");
        Console.WriteLine(newPolicy);
    }

```

```

        PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
        {
            BucketName = logBucketName,
            Policy = newPolicy,
        };
        await client.PutBucketPolicyAsync(putRequest);
        Console.WriteLine("Policy applied.");
    }

    /// <summary>
    /// This method enables logging for an Amazon S3 bucket. Logs will be
stored
    /// in the bucket you selected for logging. Selected prefix
    /// will be prepended to each log object.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client which will be
used
    /// to configure and apply logging to the selected Amazon S3 bucket.</
param>
    /// <param name="bucketName">The name of the Amazon S3 bucket for which
you
    /// wish to enable logging.</param>
    /// <param name="logBucketName">The name of the Amazon S3 bucket where
logging
    /// information will be stored.</param>
    /// <param name="logObjectKeyPrefix">The prefix to prepend to each
    /// object key.</param>
    /// <returns>Async task.</returns>
    public static async Task EnableLoggingAsync(
        IAmazonS3 client,
        string bucketName,
        string logBucketName,
        string logObjectKeyPrefix)
    {
        Console.WriteLine($"Enabling logging for bucket {bucketName}.");
        var loggingConfig = new S3BucketLoggingConfig
        {
            TargetBucketName = logBucketName,
            TargetPrefix = logObjectKeyPrefix,
        };

        var putBucketLoggingRequest = new PutBucketLoggingRequest
        {

```

```

        BucketName = bucketName,
        LoggingConfig = loggingConfig,
    };
    await client.PutBucketLoggingAsync(putBucketLoggingRequest);
    Console.WriteLine($"Logging enabled.");
}

/// <summary>
/// Loads configuration from settings files.
/// </summary>
public static void LoadConfig()
{
    _configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("settings.json") // Load settings from .json file.
        .AddJsonFile("settings.local.json", true) // Optionally, load
local settings.
        .Build();
}
}

```

- Per i dettagli sull'API, [PutBucketLogging](#) consulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Esempio 1: come impostare la registrazione delle policy di bucket

L'esempio `put-bucket-logging` seguente imposta la policy di registrazione per `amzn-s3-demo-bucket`. Innanzitutto, concedi al servizio di registrazione di log l'autorizzazione principale nella policy di bucket con il comando `put-bucket-policy`.

```

aws s3api put-bucket-policy \
    --bucket amzn-s3-demo-bucket \
    --policy file://policy.json

```

Contenuto di `policy.json`:

```
{
```



```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "S3ServerAccessLogsPolicy",
    "Effect": "Allow",
    "Principal": {"Service": "logging.s3.amazonaws.com"},
    "Action": "s3:PutObject",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/Logs/*",
    "Condition": {
      "ArnLike": {"aws:SourceARN": "arn:aws:s3:::SOURCE-BUCKET-NAME"},
      "StringEquals": {"aws:SourceAccount": "SOURCE-AWS-ACCOUNT-ID"}
    }
  }
]
```

Per applicare la policy di registrazione di log, utilizza `put-bucket-logging`.

```
aws s3api put-bucket-logging \
  --bucket amzn-s3-demo-bucket \
  --bucket-logging-status file://logging.json
```

Contenuto di `logging.json`:

```
{
  "LoggingEnabled": {
    "TargetBucket": "amzn-s3-demo-bucket",
    "TargetPrefix": "Logs/"
  }
}
```

Il comando `put-bucket-policy` è necessario per concedere le autorizzazioni `s3:PutObject` al principale del servizio di registrazione di log.

Per ulteriori informazioni, consulta [Registrazione degli accessi al server Amazon S3](#) nella Guida per l'utente di Amazon S3.

Esempio 2: come impostare una policy di bucket per la registrazione dell'accesso su un unico utente

L'esempio `put-bucket-acl` seguente imposta la policy di registrazione per `amzn-s3-demo-bucket`. L'AWS utente `bob@example.com` avrà il pieno controllo sui file di registro e nessun altro potrà accedervi. Concedi innanzitutto l'autorizzazione S3 con `put-bucket-acl`.

```
aws s3api put-bucket-acl \  
  --bucket amzn-s3-demo-bucket \  
  --grant-write URI=http://acs.amazonaws.com/groups/s3/LogDelivery \  
  --grant-read-acp URI=http://acs.amazonaws.com/groups/s3/LogDelivery
```

Applica quindi la policy di registrazione di log con `put-bucket-logging`.

```
aws s3api put-bucket-logging \  
  --bucket amzn-s3-demo-bucket \  
  --bucket-logging-status file://logging.json
```

Contenuto di `logging.json`:

```
{  
  "LoggingEnabled": {  
    "TargetBucket": "amzn-s3-demo-bucket",  
    "TargetPrefix": "amzn-s3-demo-bucket-logs/",  
    "TargetGrants": [  
      {  
        "Grantee": {  
          "Type": "AmazonCustomerByEmail",  
          "EmailAddress": "bob@example.com"  
        },  
        "Permission": "FULL_CONTROL"  
      }  
    ]  
  }  
}
```

Il comando `put-bucket-acl` è necessario per concedere al sistema di distribuzione dei log S3 le autorizzazioni necessarie (autorizzazioni di scrittura e `read-acp`).

Per ulteriori informazioni, consulta [Registrazione degli accessi al server Amazon S3](#) nella Guida per gli sviluppatori di Amazon S3.

- Per i dettagli sull'API, consulta [PutBucketLogging AWS CLI Command Reference](#).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutBucketNotification** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare PutBucketNotification.

CLI

AWS CLI

Applica una configurazione di notifica a un bucket denominato `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-notification --bucket amzn-s3-demo-bucket --notification-configuration file://notification.json
```

Il file `notification.json` è un documento JSON nella cartella corrente che specifica un argomento SNS e un tipo di evento da monitorare.

```
{
  "TopicConfiguration": {
    "Event": "s3:ObjectCreated:*",
    "Topic": "arn:aws:sns:us-west-2:123456789012:s3-notification-topic"
  }
}
```

All'argomento SNS deve essere collegata una policy IAM che consenta ad Amazon S3 di effettuarvi pubblicazioni.

```
{
  "Version": "2012-10-17",
  "Id": "example-ID",
  "Statement": [
    {
      "Sid": "example-statement-ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": [
```

```
"SNS:Publish"
],
"Resource": "arn:aws:sns:us-west-2:123456789012:amzn-s3-demo-bucket",
"Condition": {
  "ArnLike": {
    "aws:SourceArn": "arn:aws:s3:*:*:amzn-s3-demo-bucket"
  }
}
}
]
```

- Per i dettagli sull'API, consulta [PutBucketNotification AWS CLI Command Reference](#).

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo esempio configura la configurazione dell'argomento SNS per l'evento S3 ObjectRemovedDelete e abilita la notifica per il bucket s3 specificato

```
$topic = [Amazon.S3.Model.TopicConfiguration] @{
    Id = "delete-event"
    Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
    Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
$topic
```

Esempio 2: questo esempio abilita le notifiche ObjectCreatedAll relative al bucket specificato che lo invia alla funzione Lambda.

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
    Id = "ObjectCreated-Lambda"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".pem"}
            )
        }
    }
}
```

```

    )
  }
}
}

```

```

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $lambdaConfig

```

Esempio 3: questo esempio crea 2 diverse configurazioni Lambda in base a suffissi di chiave diversi e le configura entrambe in un unico comando.

```
#Lambda Config 1
```

```

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
    Id = "ObjectCreated-dada-ps1"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".ps1"}
            )
        }
    }
}

```

```
#Lambda Config 2
```

```

$secondLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = [Amazon.S3.EventType]::ObjectCreatedAll
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
    Id = "ObjectCreated-dada-json"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".json"}
            )
        }
    }
}

```

```
Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondLambdaConfig
```

- Per i dettagli sull'API, vedere [PutBucketNotification](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo esempio configura la configurazione dell'argomento SNS per l'evento S3 ObjectRemovedDelete e abilita la notifica per il bucket s3 specificato

```
$topic = [Amazon.S3.Model.TopicConfiguration] @{
    Id = "delete-event"
    Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
    Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
$topic
```

Esempio 2: questo esempio abilita le notifiche ObjectCreatedAll relative al bucket specificato che lo invia alla funzione Lambda.

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
    Id = "ObjectCreated-Lambda"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".pem"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $lambdaConfig
```

Esempio 3: questo esempio crea 2 diverse configurazioni Lambda in base a suffissi di chiave diversi e le configura entrambe in un unico comando.

```
#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
    Id = "ObjectCreated-dada-ps1"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".ps1"}
            )
        }
    }
}

#Lambda Config 2

$secondLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = [Amazon.S3.EventType]::ObjectCreatedAll
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
    Id = "ObjectCreated-dada-json"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".json"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondLambdaConfig
```

- Per i dettagli sull'API, vedere [PutBucketNotification](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutBucketNotificationConfiguration** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare `PutBucketNotificationConfiguration`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- [Elaborare le notifiche di eventi S3](#)
- [Invia notifiche di eventi a EventBridge](#)

.NET

SDK per .NET

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to enable notifications for an Amazon Simple
/// Storage Service (Amazon S3) bucket.
/// </summary>
public class EnableNotifications
{
    public static async Task Main()
    {
        const string bucketName = "amzn-s3-demo-bucket1";
        const string snsTopic = "arn:aws:sns:us-east-2:0123456789ab:bucket-
notify";
        const string sqsQueue = "arn:aws:sqs:us-
east-2:0123456789ab:Example_Queue";
```



```
IAmazonS3 client = new AmazonS3Client(Amazon.RegionEndpoint.USEast2);
await EnableNotificationAsync(client, bucketName, snsTopic,
sqsQueue);
}

/// <summary>
/// This method makes the call to the PutBucketNotificationAsync method.
/// </summary>
/// <param name="client">An initialized Amazon S3 client used to call
/// the PutBucketNotificationAsync method.</param>
/// <param name="bucketName">The name of the bucket for which
/// notifications will be turned on.</param>
/// <param name="snsTopic">The ARN for the Amazon Simple Notification
/// Service (Amazon SNS) topic associated with the S3 bucket.</param>
/// <param name="sqsQueue">The ARN of the Amazon Simple Queue Service
/// (Amazon SQS) queue to which notifications will be pushed.</param>
public static async Task EnableNotificationAsync(
    IAmazonS3 client,
    string bucketName,
    string snsTopic,
    string sqsQueue)
{
    try
    {
        // The bucket for which we are setting up notifications.
        var request = new PutBucketNotificationRequest()
        {
            BucketName = bucketName,
        };

        // Defines the topic to use when sending a notification.
        var topicConfig = new TopicConfiguration()
        {
            Events = new List<EventType> { EventType.ObjectCreatedCopy },
            Topic = snsTopic,
        };
        request.TopicConfigurations = new List<TopicConfiguration>
        {
            topicConfig,
        };
        request.QueueConfigurations = new List<QueueConfiguration>
        {
            new QueueConfiguration()
            {

```

```
        Events = new List<EventType>
        { EventType.ObjectCreatedPut },
        Queue = sqsQueue,
    },
};

// Now apply the notification settings to the bucket.
PutBucketNotificationResponse response = await
client.PutBucketNotificationAsync(request);
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error: {ex.Message}");
    }
}
```

- Per i dettagli sull'API, [PutBucketNotificationConfiguration](#) consulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Come abilitare le notifiche specificate in un bucket

L'esempio `put-bucket-notification-configuration` seguente applica una configurazione di notifica a un bucket denominato `amzn-s3-demo-bucket`. Il file `notification.json` è un documento JSON nella cartella corrente che specifica un argomento SNS e un tipo di evento da monitorare.

```
aws s3api put-bucket-notification-configuration \
    --bucket amzn-s3-demo-bucket \
    --notification-configuration file://notification.json
```

Contenuto di `notification.json`:

```
{
  "TopicConfigurations": [
    {
```

```

        "TopicArn": "arn:aws:sns:us-west-2:123456789012:s3-notification-
topic",
        "Events": [
            "s3:ObjectCreated:*"
        ]
    }
]
}

```

All'argomento SNS deve essere collegata una policy IAM che consenta ad Amazon S3 di effettuarvi pubblicazioni.

```

{
  "Version": "2012-10-17",
  "Id": "example-ID",
  "Statement": [
    {
      "Sid": "example-statement-ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": [
        "SNS:Publish"
      ],
      "Resource": "arn:aws:sns:us-west-2:123456789012::s3-notification-
topic",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:s3:*:*:amzn-s3-demo-bucket"
        }
      }
    }
  ]
}

```

- Per i dettagli sull'API, consulta [PutBucketNotificationConfiguration AWS CLI Command Reference](#).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutBucketPolicy** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare PutBucketPolicy.

C++

SDK per C++

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
bool AwsDoc::S3::putBucketPolicy(const Aws::String &bucketName,
                                const Aws::String &policyBody,
                                const Aws::S3::S3ClientConfiguration
                                &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    std::shared_ptr<Aws::StringStream> request_body =
        Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;

    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucketName);
    request.SetBody(request_body);

    Aws::S3::Model::PutBucketPolicyOutcome outcome =
        s3Client.PutBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putBucketPolicy: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Set the following policy body for the bucket '" <<
                  bucketName << "':" << std::endl << std::endl;
        std::cout << policyBody << std::endl;
    }

    return outcome.IsSuccess();
}
```

```

//! Build a policy JSON string.
/*!
    \param userArn: Aws user Amazon Resource Name (ARN).
        For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_identifiers.html#identifiers-arns.
    \param bucketName: Name of a bucket.
    \return String: Policy as JSON string.
*/

Aws::String getPolicyString(const Aws::String &userArn,
                           const Aws::String &bucketName) {
    return
        "{\n"
        "  \"Version\": \"2012-10-17\", \n"
        "  \"Statement\": [\n"
        "    {\n"
        "      \"Sid\": \"1\", \n"
        "      \"Effect\": \"Allow\", \n"
        "      \"Principal\": {\n"
        "        \"AWS\": \"\"
        + userArn +
        "\"\"\"\"      }, \n"
        "      \"Action\": [ \"s3:getObject\" ], \n"
        "      \"Resource\": [ \"arn:aws:s3:::"
        + bucketName +
        "\"/*\" ] \n"
        "    } \n"
        "  ] \n"
        "}";
}

```

- Per i dettagli sull'API, [PutBucketPolicy](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Questo esempio consente a tutti gli utenti di recuperare qualsiasi oggetto in `amzn-s3-demo-bucket` ad eccezione di quelli in `MySecretFolder`. Inoltre concede `put` l'autorizzazione all'utente `root` dell'account: `delete AWS 1234-5678-9012`

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://  
policy.json
```

policy.json:

```
{  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": "*",  
      "Action": "s3:GetObject",  
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"  
    },  
    {  
      "Effect": "Deny",  
      "Principal": "*",  
      "Action": "s3:GetObject",  
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/MySecretFolder/*"  
    },  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "AWS": "arn:aws:iam::123456789012:root"  
      },  
      "Action": [  
        "s3:DeleteObject",  
        "s3:PutObject"  
      ],  
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"  
    }  
  ]  
}
```

- Per i dettagli sull'API, consulta [PutBucketPolicy AWS CLI](#) Command Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;

import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
            <bucketName> <polFile>

        Where:
```

```

        bucketName - The Amazon S3 bucket to set the policy on.
        polFile - A JSON file containing the policy (see the Amazon S3
Readme for an example).\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String polFile = args[1];
    String policyText = getBucketPolicyFromFile(polFile);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setPolicy(s3, bucketName, policyText);
    s3.close();
}

/**
 * Sets the policy for an Amazon S3 bucket.
 *
 * @param s3          the {@link S3Client} object used to interact with the
Amazon S3 service
 * @param bucketName the name of the Amazon S3 bucket
 * @param policyText the text of the policy to be set on the bucket
 * @throws S3Exception if there is an error setting the bucket policy
 */
public static void setPolicy(S3Client s3, String bucketName, String
policyText) {
    System.out.println("Setting policy:");
    System.out.println("----");
    System.out.println(policyText);
    System.out.println("----");
    System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

    try {
        PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
            .bucket(bucketName)
            .policy(policyText)
            .build();
    }
}

```



```
s3.putBucketPolicy(policyReq);

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("Done!");
}

/**
 * Retrieves the bucket policy from a specified file.
 *
 * @param policyFile the path to the file containing the bucket policy
 * @return the content of the bucket policy file as a string
 */
public static String getBucketPolicyFromFile(String policyFile) {
    StringBuilder fileText = new StringBuilder();
    try {
        List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
        for (String line : lines) {
            fileText.append(line);
        }

    } catch (IOException e) {
        System.out.format("Problem reading file: \"%s\"", policyFile);
        System.out.println(e.getMessage());
    }

    try {
        final JsonParser parser = new
ObjectMapper().getFactory().createParser(fileText.toString());
        while (parser.nextToken() != null) {
        }

    } catch (IOException jpe) {
        jpe.printStackTrace();
    }
    return fileText.toString();
}
}
```

- Per i dettagli sull'API, [PutBucketPolicy](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Aggiungi la policy.

```
import {
  PutBucketPolicyCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Grant an IAM role GetObject access to all of the objects
 * in the provided bucket.
 * @param {{ bucketName: string, iamRoleArn: string }}
 */
export const main = async ({ bucketName, iamRoleArn }) => {
  const client = new S3Client({});
  const command = new PutBucketPolicyCommand({
    // This is a resource-based policy. For more information on resource-based
    // policies,
    // see https://docs.aws.amazon.com/IAM/latest/UserGuide/
    // access_policies.html#policies_resource-based.
    Policy: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Principal: {
            AWS: iamRoleArn,
          },
        },
      ],
    }),
  });
```

```

        Action: "s3:GetObject",
        Resource: `arn:aws:s3:::${bucketName}/*`,
    },
],
}),
// Apply the preceding policy to this bucket.
Bucket: bucketName,
});

try {
    await client.send(command);
    console.log(
        `GetObject access to the bucket "${bucketName}" was granted to the provided
IAM role.`,
    );
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "MalformedPolicy"
    ) {
        console.error(
            `Error from S3 while setting the bucket policy for the bucket
"${bucketName}". The policy was malformed.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while setting the bucket policy for the bucket
"${bucketName}". ${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [PutBucketPolicy](#) consulta AWS SDK per JavaScript API Reference.

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def put_policy(self, policy):
        """
        Apply a security policy to the bucket. Policies control users' ability
        to perform specific actions, such as listing the objects in the bucket.

        :param policy: The policy to apply to the bucket.
        """
        try:
            self.bucket.Policy().put(Policy=json.dumps(policy))
            logger.info("Put policy %s for bucket '%s'.", policy,
self.bucket.name)
        except ClientError:
            logger.exception("Couldn't apply policy to bucket '%s'.",
self.bucket.name)
            raise
```

- Per i dettagli sull'API, consulta [PutBucketPolicy AWS SDK for Python \(Boto3\) API Reference](#).

Ruby

SDK per Ruby

Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
  configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Sets a policy on a bucket.
  #
  def policy(policy)
    @bucket_policy.put(policy: policy)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
    #{e.message}"
    false
  end
end
```

- Per i dettagli sull'API, [PutBucketPolicy](#) consulta AWS SDK per Ruby API Reference.

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.  
    " Example policy JSON string  
    " iv_policy = '{"Version":"2012-10-17",          "Statement":  
[{"Effect":"Allow","Principal":{"AWS":"arn:aws:iam::123456789012:user/  
user"},"Action":["s3:GetObject"],"Resource":["arn:aws:s3:::bucketname/*"]}]}'  
    lo_s3->putbucketpolicy(  
        iv_bucket = iv_bucket_name  
        iv_policy = iv_policy ).  
    MESSAGE 'Bucket policy set.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Per i dettagli sulle API, [PutBucketPolicy](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutBucketReplication** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare PutBucketReplication.

CLI

AWS CLI

Come configurare la replica per un bucket S3

L'esempio put-bucket-replication seguente applica una configurazione di replica al bucket S3 specificato.

```
aws s3api put-bucket-replication \  
  --bucket amzn-s3-demo-bucket1 \  
  --replication-configuration file://replication.json
```

Contenuto di replication.json:

```
{  
  "Role": "arn:aws:iam::123456789012:role/s3-replication-role",  
  "Rules": [  
    {  
      "Status": "Enabled",  
      "Priority": 1,  
      "DeleteMarkerReplication": { "Status": "Disabled" },  
      "Filter" : { "Prefix": ""},  
      "Destination": {  
        "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket2"  
      }  
    }  
  ]  
}
```

Il bucket di destinazione deve avere il controllo delle versioni abilitato. Il ruolo specificato deve disporre dell'autorizzazione di scrittura nel bucket di destinazione e di una relazione di attendibilità che consenta ad Amazon S3 di assumere il ruolo.

Esempio di policy di autorizzazione di un ruolo:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:GetReplicationConfiguration",  
        "s3:ListBucket"  
      ],  
      "Resource": [  
        "arn:aws:s3:::amzn-s3-demo-bucket1"  
      ]  
    }  
  ]  
}
```

```

    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObjectVersion",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateDelete",
        "s3:ReplicateTags"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket2/*"
    }
  ]
}

```

Esempio di policy delle relazioni di attendibilità:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

Questo comando non produce alcun output.

Per ulteriori informazioni, consulta il [seguente argomento](#) nella Guida per l'utente della console Amazon Simple Storage Service.

- Per i dettagli sull'API, consulta [PutBucketReplication AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 * Sets the replication configuration for an Amazon S3 bucket.
 *
 * @param s3Client          the S3Client instance to use for the operation
 * @param sourceBucketName the name of the source bucket
 * @param destBucketName   the name of the destination bucket
 * @param destinationBucketARN the Amazon Resource Name (ARN) of the
destination bucket
 * @param roleARN          the ARN of the IAM role to use for the
replication configuration
 */
public static void setReplication(S3Client s3Client, String sourceBucketName,
String destBucketName, String destinationBucketARN, String roleARN) {
    try {
        Destination destination = Destination.builder()
            .bucket(destinationBucketARN)
            .storageClass(StorageClass.STANDARD)
            .build();

        // Define a prefix filter for replication.
        ReplicationRuleFilter ruleFilter = ReplicationRuleFilter.builder()
            .prefix("documents/")
            .build();

        // Define delete marker replication setting.
        DeleteMarkerReplication deleteMarkerReplication =
DeleteMarkerReplication.builder()
            .status(DeleteMarkerReplicationStatus.DISABLED)
            .build();
```

```
// Create the replication rule.
ReplicationRule replicationRule = ReplicationRule.builder()
    .priority(1)
    .filter(ruleFilter)
    .status(ReplicationRuleStatus.ENABLED)
    .deleteMarkerReplication(deleteMarkerReplication)
    .destination(destination)
    .build();

List<ReplicationRule> replicationRuleList = new ArrayList<>();
replicationRuleList.add(replicationRule);

// Define the replication configuration with IAM role.
ReplicationConfiguration configuration =
ReplicationConfiguration.builder()
    .role(roleARN)
    .rules(replicationRuleList)
    .build();

// Apply the replication configuration to the source bucket.
PutBucketReplicationRequest replicationRequest =
PutBucketReplicationRequest.builder()
    .bucket(sourceBucketName)
    .replicationConfiguration(configuration)
    .build();

s3Client.putBucketReplication(replicationRequest);
System.out.println("Replication configuration set successfully.");

} catch (IllegalArgumentException e) {
    System.err.println("Configuration error: " + e.getMessage());
} catch (S3Exception e) {
    System.err.println("S3 Exception: " +
e.awsErrorDetails().errorMessage());
    System.err.println("Status Code: " + e.statusCode());
    System.err.println("Error Code: " + e.awsErrorDetails().errorCode());

} catch (SdkException e) {
    System.err.println("SDK Exception: " + e.getMessage());
}
}
```

- Per i dettagli sull'API, [PutBucketReplication](#) consulta AWS SDK for Java 2.x API Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: Questo esempio imposta una configurazione di replica con un'unica regola che consente la replica nel bucket 'amzn-s3-demo-bucket' di tutti i nuovi oggetti creati con il prefisso del nome chiave "" nel bucket 'amzn-s3-demo-bucket'. TaxDocs

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

Esempio 2: Questo esempio imposta una configurazione di replica con più regole che consentono la replica nel bucket 'amzn-s3-demo-bucket' di qualsiasi nuovo oggetto creato con il prefisso del nome chiave "TaxDocs" o OtherDocs "». I prefissi di chiave non devono sovrapporsi.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
```

```

$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params

```

Esempio 3: Questo esempio aggiorna la configurazione di replica sul bucket specificato per disabilitare la regola che controlla la replica degli oggetti con il prefisso del nome chiave TaxDocs "" nel bucket 'amzn-s3-demo-bucket'.

```

$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params

```

- Per [PutBucketReplication AWS Strumenti per PowerShell](#) dettagli sull'API, vedere in Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: Questo esempio imposta una configurazione di replica con un'unica regola che consente la replica nel bucket 'amzn-s3-demo-bucket' di tutti i nuovi oggetti creati con il prefisso del nome chiave "" nel bucket 'amzn-s3-demo-bucket'. TaxDocs

```

$rule1 = New-Object Amazon.S3.Model.ReplicationRule

```

```

$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params

```

Esempio 2: Questo esempio imposta una configurazione di replica con più regole che consentono la replica nel bucket 'amzn-s3-demo-bucket' di qualsiasi nuovo oggetto creato con il prefisso del nome chiave "TaxDocs" o OtherDocs "». I prefissi di chiave non devono sovrapporsi.

```

$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params

```

Esempio 3: Questo esempio aggiorna la configurazione di replica sul bucket specificato per disabilitare la regola che controlla la replica degli oggetti con il prefisso del nome chiave TaxDocs "" nel bucket 'amzn-s3-demo-bucket'.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- Per [PutBucketReplication AWS Strumenti per PowerShell](#) dettagli sull'API, vedere in Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutBucketRequestPayment** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare PutBucketRequestPayment.

CLI

AWS CLI

Esempio 1: come abilitare la configurazione "requester pays" per un bucket

L'esempio put-bucket-request-payment seguente abilita requester pays per il bucket specificato.

```
aws s3api put-bucket-request-payment \
```

```
--bucket amzn-s3-demo-bucket \  
--request-payment-configuration '{"Payer": "Requester"}'
```

Questo comando non produce alcun output.

Esempio 2: come disabilitare la configurazione “requester pays” per un bucket

L'esempio `put-bucket-request-payment` seguente disabilita `requester pays` per il bucket specificato.

```
aws s3api put-bucket-request-payment \  
--bucket amzn-s3-demo-bucket \  
--request-payment-configuration '{"Payer": "BucketOwner"}'
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [PutBucketRequestPayment AWS CLI Command Reference](#).

PowerShell

Strumenti per PowerShell V4

Esempio 1: aggiorna la configurazione del pagamento della richiesta per il bucket denominato “amzn-s3-demo-bucket” in modo che alla persona che richiede il download dal bucket venga addebitato il costo del download. Per impostazione predefinita, il proprietario del bucket paga i download. Per riportare la richiesta di pagamento ai valori predefiniti, usa 'BucketOwner' per il parametro `RequestPaymentConfiguration_Payer`.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- Per i dettagli sull'API, vedere [PutBucketRequestPayment](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: aggiorna la configurazione del pagamento della richiesta per il bucket denominato “amzn-s3-demo-bucket” in modo che alla persona che richiede il download dal bucket venga addebitato il costo del download. Per impostazione predefinita, il proprietario del bucket paga i download. Per riportare la richiesta di pagamento ai valori predefiniti, usa 'BucketOwner' per il parametro `RequestPaymentConfiguration_Payer`.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- Per i dettagli sull'API, vedere [PutBucketRequestPayment](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutBucketTagging** con una CLI

Gli esempi di codice seguenti mostrano come utilizzare PutBucketTagging.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Guida introduttiva allo storage di oggetti](#)

CLI

AWS CLI

Il comando seguente applica una configurazione di tagging da un bucket denominato amzn-s3-demo-bucket.

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging file://  
tagging.json
```

Il file `tagging.json` è un documento JSON nella cartella corrente in cui sono specificati i tag.

```
{  
  "TagSet": [  
    {  
      "Key": "organization",  
      "Value": "marketing"  
    }  
  ]  
}
```


In alternativa, applica una configurazione di tagging ad `amzn-s3-demo-bucket` direttamente dalla riga di comando.

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging  
'TagSet=[{Key=organization,Value=marketing}]'
```

- Per i dettagli sull'API, consulta [PutBucketTagging AWS CLI Command Reference](#).

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando applica due tag a un bucket denominato **cloudtrail-test-2018**, ovvero un tag con una chiave Stage e un valore Test e un tag con una chiave Environment e un valore Alpha. Per verificare che i tag siano stati aggiunti al bucket, esegui **Get-S3BucketTagging -BucketName bucket_name**. I risultati dovrebbero riportare i tag applicati al bucket nel primo comando. Nota che **Write-S3BucketTagging** sovrascrive l'intero set di tag esistente in un bucket. Per aggiungere o eliminare singoli tag, esegui i cmdlet **Add-RGResourceTag** e **Remove-RGResourceTag** delle API Gruppi di risorse e Tagging. In alternativa, utilizza Tag Editor nella console di AWS gestione per gestire i tag dei bucket S3.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";  
Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Esempio 2: questo comando reindirizza un bucket denominato **cloudtrail-test-2018** nel cmdlet **Write-S3BucketTagging**. Applica i tag Stage:Production e Department:Finance al bucket. Nota che **Write-S3BucketTagging** sovrascrive l'intero set di tag esistente in un bucket.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging  
-TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";  
Value="Finance" } )
```

- Per i dettagli sull'API, vedere [PutBucketTagging](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando applica due tag a un bucket denominato **cloudtrail-test-2018**, ovvero un tag con una chiave Stage e un valore Test e un tag con una chiave

Environment e un valore Alpha. Per verificare che i tag siano stati aggiunti al bucket, esegui **Get-S3BucketTagging -BucketName bucket_name**. I risultati dovrebbero riportare i tag applicati al bucket nel primo comando. Nota che **Write-S3BucketTagging** sovrascrive l'intero set di tag esistente in un bucket. Per aggiungere o eliminare singoli tag, esegui i cmdlet **Add-RGTResourceTag** e **Remove-RGTResourceTag** delle API Gruppi di risorse e Tagging. In alternativa, utilizza Tag Editor nella console di AWS gestione per gestire i tag dei bucket S3.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage"; Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Esempio 2: questo comando reindirizza un bucket denominato **cloudtrail-test-2018** nel cmdlet **Write-S3BucketTagging**. Applica i tag Stage:Production e Department:Finance al bucket. Nota che **Write-S3BucketTagging** sovrascrive l'intero set di tag esistente in un bucket.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging -TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department"; Value="Finance" } )
```

- Per i dettagli sull'API, vedere [PutBucketTagging](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, vedere. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutBucketVersioning** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare PutBucketVersioning.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Guida introduttiva allo storage di oggetti](#)

CLI

AWS CLI

Il comando seguente abilita il controllo delle versioni sul bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-configuration Status=Enabled
```

Il comando seguente abilita il controllo delle versioni e utilizza una modalità di autenticazione a più fattori (MFA).

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-configuration Status=Enabled --mfa "SERIAL 123456"
```

- Per i dettagli sull'API, consulta [PutBucketVersioning AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 * Enables bucket versioning for the specified S3 bucket.
 *
 * @param s3Client the S3 client to use for the operation
 * @param bucketName the name of the S3 bucket to enable versioning for
 */
public static void enableBucketVersioning(S3Client s3Client, String
bucketName){
    VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
        .status(BucketVersioningStatus.ENABLED)
```

```
        .build();

        PutBucketVersioningRequest versioningRequest =
        PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        s3Client.putBucketVersioning(versioningRequest);
        System.out.println("Bucket versioning has been enabled for "+bucketName);
    }
}
```

- Per i dettagli sull'API, [PutBucketVersioning](#) consulta AWS SDK for Java 2.x API Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: il comando abilita il controllo delle versioni per il bucket S3 specificato.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -
VersioningConfig_Status Enabled
```

- Per i dettagli sull'API, vedere [PutBucketVersioning](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: il comando abilita il controllo delle versioni per il bucket S3 specificato.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -
VersioningConfig_Status Enabled
```

- Per i dettagli sull'API, vedere [PutBucketVersioning](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.  
  " Example: Enable versioning on a bucket  
  " iv_status = 'Enabled'  
  lo_s3->putbucketversioning(  
    iv_bucket = iv_bucket_name  
    io_versioningconfiguration = NEW /aws1/cl_s3_versioningconf(  
      iv_status = iv_status ) ).  
  MESSAGE 'Bucket versioning enabled.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
  MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Per i dettagli sulle API, [PutBucketVersioning](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutBucketWebsite** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare PutBucketWebsite.

.NET

SDK per .NET

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Put the website configuration.
PutBucketWebsiteRequest putRequest = new
PutBucketWebsiteRequest()
{
    BucketName = bucketName,
    WebsiteConfiguration = new WebsiteConfiguration()
    {
        IndexDocumentSuffix = indexDocumentSuffix,
        ErrorDocument = errorDocument,
    },
};
PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);
```

- Per i dettagli sull'API, [PutBucketWebsite](#) consulta AWS SDK per .NET API Reference.

C++

SDK per C++

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
```

```

        const Aws::String &indexPage, const Aws::String
&errorPage,
        const Aws::S3::S3ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPage);

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
        client.PutBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
            << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Success: Set website configuration for bucket '"
            << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

```

- Per i dettagli sull'API, [PutBucketWebsite](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Applica una configurazione statica del sito web a un bucket denominato `amzn-s3-demo-bucket`.

```
aws s3api put-bucket-website --bucket amzn-s3-demo-bucket --website-configuration file://website.json
```

Il file `website.json` è un documento JSON nella cartella corrente che specifica le pagine dell'indice e degli errori per il sito web:

```
{
  "IndexDocument": {
    "Suffix": "index.html"
  },
  "ErrorDocument": {
    "Key": "error.html"
  }
}
```

- Per i dettagli sull'API, consulta [PutBucketWebsite AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
```



```

* <p>
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucketName> [indexdoc]\s

            Where:
                bucketName    - The Amazon S3 bucket to set the website
configuration on.\s
                indexdoc - The index document, ex. 'index.html'
                        If not specified, 'index.html' will be set.

            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String indexDoc = "index.html";
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setWebsiteConfig(s3, bucketName, indexDoc);
        s3.close();
    }

    /**
     * Sets the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance to use for the AWS SDK operations.
     * @param bucketName The name of the S3 bucket to configure.
     * @param indexDoc The name of the index document to use for the website
configuration.
     */
    public static void setWebsiteConfig(S3Client s3, String bucketName, String
indexDoc) {

```

```
try {
    WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
        .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
        .build();

    PutBucketWebsiteRequest pubWebsiteReq =
    PutBucketWebsiteRequest.builder()
        .bucket(bucketName)
        .websiteConfiguration(websiteConfig)
        .build();

    s3.putBucketWebsite(pubWebsiteReq);
    System.out.println("The call was successful");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Per i dettagli sull'API, [PutBucketWebsite](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Imposta la configurazione del sito web.

```
import {
    PutBucketWebsiteCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";
```

```
/**
 * Configure an Amazon S3 bucket to serve a static website.
 * Website access must also be granted separately. For more information
 * on setting the permissions for website access, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/WebsiteAccessPermissionsReqd.html.
 *
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new PutBucketWebsiteCommand({
    Bucket: bucketName,
    WebsiteConfiguration: {
      ErrorDocument: {
        // The object key name to use when a 4XX class error occurs.
        Key: "error.html",
      },
      IndexDocument: {
        // A suffix that is appended to a request when the request is
        // for a directory.
        Suffix: "index.html",
      },
    },
  });

  try {
    await client.send(command);
    console.log(
      `The bucket "${bucketName}" has been configured as a static website.`
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while configuring the bucket "${bucketName}" as a static website. The bucket doesn't exist.`
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while configuring the bucket "${bucketName}" as a static website. ${caught.name}: ${caught.message}`
      );
    }
  }
}
```

```
    );  
  } else {  
    throw caught;  
  }  
}  
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [PutBucketWebsite](#) consulta AWS SDK per JavaScript API Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: il comando abilita l'hosting del sito web per il bucket specificato con il documento dell'indice denominato "index.html" e il documento degli errori denominato "error.html".

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'  
  -WebsiteConfiguration_IndexDocumentSuffix 'index.html' -  
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Per i dettagli sull'API, vedere [PutBucketWebsite](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: il comando abilita l'hosting del sito web per il bucket specificato con il documento dell'indice denominato "index.html" e il documento degli errori denominato "error.html".

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'  
  -WebsiteConfiguration_IndexDocumentSuffix 'index.html' -  
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Per i dettagli sull'API, vedere [PutBucketWebsite](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Ruby

SDK per Ruby

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket website actions.
class BucketWebsiteWrapper
  attr_reader :bucket_website

  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
  # configured with an existing bucket.
  def initialize(bucket_website)
    @bucket_website = bucket_website
  end

  # Sets a bucket as a static website.
  #
  # @param index_document [String] The name of the index document for the
  # website.
  # @param error_document [String] The name of the error document to show for 4XX
  # errors.
  # @return [Boolean] True when the bucket is configured as a website; otherwise,
  # false.
  def set_website(index_document, error_document)
    @bucket_website.put(
      website_configuration: {
        index_document: { suffix: index_document },
        error_document: { key: error_document }
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
    why: #{e.message}"
    false
  end
end
```

```
end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  index_document = "index.html"
  error_document = "404.html"

  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))
  return unless wrapper.set_website(index_document, error_document)

  puts "Successfully configured bucket #{bucket_name} as a static website."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, [PutBucketWebsite](#) consulta AWS SDK per Ruby API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutObject** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare PutObject.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- [Informazioni di base](#)
- [Guida introduttiva allo storage di oggetti](#)
- [Eseguire richieste condizionali](#)
- [Monitorare caricamenti e scaricamenti](#)
- [Utilizzare la funzionalità per l'integrità degli oggetti di Amazon S3](#)

.NET

SDK per .NET (v4)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>
/// Shows how to upload a file from the local computer to an Amazon S3
/// bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to which the object
/// will be uploaded.</param>
/// <param name="objectName">The object to upload.</param>
/// <param name="filePath">The path, including file name, of the object
/// on the local computer to upload.</param>
/// <returns>A boolean value indicating the success or failure of the
/// upload procedure.</returns>
public async Task<bool> UploadFileAsync(
    string bucketName,
    string objectName,
    string filePath)
{
    try
    {
        var request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
            FilePath = filePath,
        };

        var response = await _amazonS3.PutObjectAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error uploading {objectName}: {ex.Message}");
    }
}
```

```
        return false;
    }
}
```

- Per i dettagli sull'API, [PutObject](#) consulta AWS SDK per .NET API Reference.

SDK per .NET

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Caricare un oggetto con la crittografia lato server.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to upload an object to an Amazon Simple Storage
/// Service (Amazon S3) bucket with server-side encryption enabled.
/// </summary>
public class ServerSideEncryption
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "samplefile.txt";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2.
        IAmazonS3 client = new AmazonS3Client();

        await WritingAnObjectAsync(client, bucketName, keyName);
    }
}
```



```
/// <summary>
/// Upload a sample object include a setting for encryption.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
/// to upload a file and apply server-side encryption.</param>
/// <param name="bucketName">The name of the Amazon S3 bucket where the
/// encrypted object will reside.</param>
/// <param name="keyName">The name for the object that you want to
/// create in the supplied bucket.</param>
public static async Task WritingAnObjectAsync(IAmazonS3 client, string
bucketName, string keyName)
{
    try
    {
        var putRequest = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = keyName,
            ContentBody = "sample text",
            ServerSideEncryptionMethod =
ServerSideEncryptionMethod.AES256,
        };

        var putResponse = await client.PutObjectAsync(putRequest);

        // Determine the encryption state of an object.
        GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
        {
            BucketName = bucketName,
            Key = keyName,
        };
        GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);
        ServerSideEncryptionMethod objectEncryption =
response.ServerSideEncryptionMethod;

        Console.WriteLine($"Encryption method used: {0}",
objectEncryption.ToString());
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error: '{ex.Message}' when writing an
object");
    }
}
```

```

    }
  }
}

```

Inserisce un oggetto utilizzando una richiesta condizionale.

```

/// <summary>
/// Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite using an IfNoneMatch condition for the object key.
/// </summary>
/// <param name="objectKey">The key of the object to upload.</param>
/// <param name="bucket">The source bucket of the object.</param>
/// <param name="content">The content to upload as a string.</param>
/// <returns>The ETag if the conditional write is successful, empty
otherwise.</returns>
public async Task<string> PutObjectConditional(string objectKey, string
bucket, string content)
{
    try
    {
        var putObjectRequest = new PutObjectRequest
        {
            BucketName = bucket,
            Key = objectKey,
            ContentBody = content,
            IfNoneMatch = "*"
        };

        var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
        _logger.LogInformation($"Conditional write successful for key
{objectKey} in bucket {bucket}.");
        return putResult.ETag;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional write failed: Precondition
failed");
        }
        else
    }
}

```

```

        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return string.Empty;
    }
}

```

- Per i dettagli sull'API, [PutObject](#) consulta AWS SDK per .NET API Reference.

Bash

AWS CLI con lo script Bash

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file to.
#     $2 - The path and file name of the local file to copy to the bucket.
#     $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#     0 - If successful.

```

```
#      1 - If it fails.
#####
function copy_file_to_bucket() {
    local response bucket_name source_file destination_file_name
    bucket_name=$1
    source_file=$2
    destination_file_name=$3

    response=$(aws s3api put-object \
        --bucket "$bucket_name" \
        --body "$source_file" \
        --key "$destination_file_name")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "ERROR: AWS reports put-object operation failed.\n$response"
        return 1
    fi
}
```

- Per i dettagli sull'API, consulta [PutObject AWS CLI](#) Command Reference.

C++

SDK per C++

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
bool AwsDoc::S3::putObject(const Aws::String &bucketName,
                           const Aws::String &fileName,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    //We are using the name of the file as the key for the object in the bucket.
```

```

//However, this is just a string and can be set according to your retrieval
needs.
request.SetKey(fileName);

std::shared_ptr<Aws::IOStream> inputData =
    Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                   fileName.c_str(),
                                   std::ios_base::in |
std::ios_base::binary);

if (!*inputData) {
    std::cerr << "Error unable to read file " << fileName << std::endl;
    return false;
}

request.SetBody(inputData);

Aws::S3::Model::PutObjectOutcome outcome =
    s3Client.PutObject(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error: putObject: " <<
        outcome.GetError().GetMessage() << std::endl;
} else {
    std::cout << "Added object '" << fileName << "' to bucket '"
        << bucketName << "'.";
}

return outcome.IsSuccess();
}

```

- Per i dettagli sull'API, [PutObject](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Esempio 1: carica un oggetto in Amazon S3

Il comando `put-object` seguente carica un oggetto in Amazon S3.

```
aws s3api put-object \
```

```
--bucket amzn-s3-demo-bucket \  
--key my-dir/MySampleImage.png \  
--body MySampleImage.png
```

Per ulteriori informazioni sul caricamento di oggetti, consulta Uploading Objects < <http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html> > nella Amazon S3 Developer Guide.

Esempio 2: carica un file video in Amazon S3

Il comando `put-object` seguente carica un file video.

```
aws s3api put-object \  
  --bucket amzn-s3-demo-bucket \  
  --key my-dir/big-video-file.mp4 \  
  --body /media/videos/f-sharp-3-data-services.mp4
```

Per ulteriori informazioni sul caricamento di oggetti, consulta Uploading Objects < <http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html> > nella Amazon S3 Developer Guide.

- Per i dettagli sull'API, consulta Command [PutObject](#)Reference AWS CLI .

Go

SDK per Go V2

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Inserisce un oggetto in un bucket utilizzando l'API di basso livello.

```
import (  
  "bytes"  
  "context"  
  "errors"  
  "fmt"
```

```
"io"
"log"
"os"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    file, err := os.Open(fileName)
    if err != nil {
        log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
    } else {
        defer file.Close()
        _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
            Bucket: aws.String(bucketName),
            Key:    aws.String(objectKey),
            Body:   file,
        })
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
                log.Printf("Error while uploading object to %s. The object is too large.\n"+
                    "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
                    "or the multipart upload API (5TB max).", bucketName)
            } else {
```

```
    log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
        fileName, bucketName, objectKey, err)
}
} else {
    err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
        ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
aws.String(objectKey)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
    }
}
}
return err
}
```

Carica un oggetto in un bucket utilizzando un gestore di trasferimento.

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}
```



```
// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
string, contents string) (string, error) {
    var outKey string
    input := &s3.PutObjectInput{
        Bucket:      aws.String(bucket),
        Key:         aws.String(key),
        Body:        bytes.NewReader([]byte(contents)),
        ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
    }
    output, err := actor.S3Manager.Upload(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    } else {
        err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
            Bucket: aws.String(bucket),
            Key:    aws.String(key),
        }, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
            bucket)
        } else {
            outKey = *output.Key
        }
    }
    return outKey, err
}
```

- Per i dettagli sull'API, [PutObject](#) consulta AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Carica un file in un bucket utilizzando un'interfaccia [S3Client](#).

```
/**
 * Uploads a local file to an AWS S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to upload the file to
 * @param key         the key (object name) to use for the uploaded file
 * @param objectPath  the local file path of the file to be uploaded
 * @return a {@link CompletableFuture} that completes with the {@link
 * PutObjectResponse} when the upload is successful, or throws a {@link
 * RuntimeException} if the upload fails
 */
public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
    return response.whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to upload file", ex);
        }
    });
}
```

Usa un [S3TransferManager](#) per [caricare un file](#) in un bucket. Visualizza il [file completo](#) ed esegui il [test](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

    public String uploadFile(S3TransferManager transferManager, String
bucketName,

                            String key, URI filePathURI) {
        UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
            .putObjectRequest(b -> b.bucket(bucketName).key(key))
            .source(Paths.get(filePathURI))
            .build();

        FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

        CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
        return uploadResult.response().eTag();
    }
```

Caricare un oggetto in un bucket e impostare tag mediante un'interfaccia [S3Client](#).

```
/**
 * Puts tags on an Amazon S3 object.
 *
 * @param s3 An {@link S3Client} object that represents the Amazon S3 client.
 * @param bucketName The name of the Amazon S3 bucket.
 * @param objectKey The key of the Amazon S3 object.
 * @param objectPath The file path of the object to be uploaded.
 */
public static void putS3ObjectTags(S3Client s3, String bucketName, String
objectKey, String objectPath) {
```

```
try {
    Tag tag1 = Tag.builder()
        .key("Tag 1")
        .value("This is tag 1")
        .build();

    Tag tag2 = Tag.builder()
        .key("Tag 2")
        .value("This is tag 2")
        .build();

    List<Tag> tags = new ArrayList<>();
    tags.add(tag1);
    tags.add(tag2);

    Tagging allTags = Tagging.builder()
        .tagSet(tags)
        .build();

    PutObjectRequest putOb = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .tagging(allTags)
        .build();

    s3.putObject(putOb,
        RequestBody.fromBytes(getObjectFile(objectPath)));

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

/**
 * Updates the tags associated with an object in an Amazon S3 bucket.
 *
 * @param s3 an instance of the S3Client class, which is used to interact
with the Amazon S3 service
 * @param bucketName the name of the S3 bucket containing the object
 * @param objectKey the key (or name) of the object in the S3 bucket
 * @throws S3Exception if there is an error updating the object's tags
 */
```

```
public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
    try {
        GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
        List<Tag> obTags = getTaggingRes.tagSet();
        for (Tag sinTag : obTags) {
            System.out.println("The tag key is: " + sinTag.key());
            System.out.println("The tag value is: " + sinTag.value());
        }

        // Replace the object's tags with two new tags.
        Tag tag3 = Tag.builder()
            .key("Tag 3")
            .value("This is tag 3")
            .build();

        Tag tag4 = Tag.builder()
            .key("Tag 4")
            .value("This is tag 4")
            .build();

        List<Tag> tags = new ArrayList<>();
        tags.add(tag3);
        tags.add(tag4);

        Tagging updatedTags = Tagging.builder()
            .tagSet(tags)
            .build();

        PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .tagging(updatedTags)
            .build();

        s3.putObjectTagging(taggingRequest1);
    }
}
```

```
        GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
        List<Tag> modTags = getTaggingRes2.tagSet();
        for (Tag sinTag : modTags) {
            System.out.println("The tag key is: " + sinTag.key());
            System.out.println("The tag value is: " + sinTag.value());
        }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

/**
 * Retrieves the contents of a file as a byte array.
 *
 * @param filePath the path of the file to be read
 * @return a byte array containing the contents of the file, or null if an
error occurs
 */
private static byte[] getObjectFile(String filePath) {
    FileInputStream fileInputStream = null;
    byte[] byteArray = null;

    try {
        File file = new File(filePath);
        byteArray = new byte[(int) file.length()];
        fileInputStream = new FileInputStream(file);
        fileInputStream.read(byteArray);

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    return byteArray;
}
```

```
}  
}
```

Caricare un oggetto in un bucket e impostare metadati mediante un'interfaccia [S3Client](#).

```
import software.amazon.awssdk.core.sync.RequestBody;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.PutObjectRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
  
import java.io.File;  
import java.util.HashMap;  
import java.util.Map;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 * <p>  
 * For more information, see the following documentation topic:  
 * <p>  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
public class PutObjectMetadata {  
    public static void main(String[] args) {  
        final String USAGE = ""  
  
            Usage:  
            <bucketName> <objectKey> <objectPath>\s  
  
            Where:  
            bucketName - The Amazon S3 bucket to upload an object into.  
            objectKey - The object to upload (for example, book.pdf).  
            objectPath - The path where the file is located (for example, C:/  
AWS/book2.pdf).\s  
            "";  
  
        if (args.length != 3) {  
            System.out.println(USAGE);  
            System.exit(1);  
        }  
    }  
}
```

```
    }

    String bucketName = args[0];
    String objectKey = args[1];
    String objectPath = args[2];
    System.out.println("Putting object " + objectKey + " into bucket " +
bucketName);
    System.out.println("  in bucket: " + bucketName);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    putS3Object(s3, bucketName, objectKey, objectPath);
    s3.close();
}

/**
 * Uploads an object to an Amazon S3 bucket with metadata.
 *
 * @param s3 the S3Client object used to interact with the Amazon S3 service
 * @param bucketName the name of the S3 bucket to upload the object to
 * @param objectKey the name of the object to be uploaded
 * @param objectPath the local file path of the object to be uploaded
 */
public static void putS3Object(S3Client s3, String bucketName, String
objectKey, String objectPath) {
    try {
        Map<String, String> metadata = new HashMap<>();
        metadata.put("author", "Mary Doe");
        metadata.put("version", "1.0.0.0");

        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .metadata(metadata)
            .build();

        s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
        System.out.println("Successfully placed " + objectKey + " into bucket
" + bucketName);

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
    }
}
```



```
        System.exit(1);
    }
}
}
```

Caricare un oggetto in un bucket e impostare un valore di conservazione per l'oggetto mediante un'interfaccia [S3Client](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */

public class PutObjectRetention {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <key> <bucketName>\s

            Where:
                key - The name of the object (for example, book.pdf).\s
                bucketName - The Amazon S3 bucket name that contains the object
                (for example, bucket1).\s
            """;
    }
}
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String key = args[0];
        String bucketName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setRentionPeriod(s3, key, bucketName);
        s3.close();
    }

    /**
     * Sets the retention period for an object in an Amazon S3 bucket.
     *
     * @param s3      the S3Client object used to interact with the Amazon S3
     service
     * @param key     the key (name) of the object in the S3 bucket
     * @param bucket  the name of the S3 bucket where the object is stored
     *
     * @throws S3Exception if an error occurs while setting the object retention
     period
     */
    public static void setRentionPeriod(S3Client s3, String key, String bucket) {
        try {
            LocalDate localDate = LocalDate.parse("2020-07-17");
            LocalDateTime localDateTime = localDate.atStartOfDay();
            Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

            ObjectLockRetention lockRetention = ObjectLockRetention.builder()
                .mode("COMPLIANCE")
                .retainUntilDate(instant)
                .build();

            PutObjectRetentionRequest retentionRequest =
                PutObjectRetentionRequest.builder()
                    .bucket(bucket)
                    .key(key)
                    .bypassGovernanceRetention(true)
```

```
        .retention(lockRetention)
        .build();

    // To set Retention on an object, the Amazon S3 bucket must support
    object
    // locking, otherwise an exception is thrown.
    s3.putObjectRetention(retentionRequest);
    System.out.print("An object retention configuration was successfully
    placed on the object");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Per i dettagli sull'API, consulta la sezione [PutObject AWS SDK for Java 2.x API Reference](#).

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Carica l'oggetto.

```
import { readFile } from "node:fs/promises";

import {
    PutObjectCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Upload a file to an S3 bucket.
```

```

* @param {{ bucketName: string, key: string, filePath: string }}
*/
export const main = async ({ bucketName, key, filePath }) => {
  const client = new S3Client({});
  const command = new PutObjectCommand({
    Bucket: bucketName,
    Key: key,
    Body: await readFile(filePath),
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "EntityTooLarge"
    ) {
      console.error(
        `Error from S3 while uploading object to ${bucketName}. \
The object was too large. To upload objects larger than 5GB, use the S3 console \
(160GB max) \
or the multipart upload API (5TB max).`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while uploading object to ${bucketName}. ${caught.name}: \
${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

```

Carica l'oggetto a condizione che il relativo ETag corrisponda a quello fornito.

```

import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,

```

```
S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
```

```
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).
- Per i dettagli sull'API, [PutObject](#) consulta AWS SDK per JavaScript API Reference.

Kotlin

SDK per Kotlin

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
suspend fun putS3Object(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            body = File(objectPath).asByteStream()
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}
```

- Per i dettagli sull'API, [PutObject](#) consulta AWS SDK for Kotlin API reference.

PHP

SDK per PHP

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Carica un oggetto in un bucket.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

$fileName = __DIR__ . "/local-file-" . uniqid();
try {
    $this->s3client->putObject([
        'Bucket' => $this->bucketName,
        'Key' => $fileName,
        'SourceFile' => __DIR__ . '/testfile.txt'
    ]);
    echo "Uploaded $fileName to $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to upload $fileName with error: " . $exception-
    >getMessage();
    exit("Please fix error with file upload before continuing.");
}
```

- Per i dettagli sull'API, [PutObject](#) consulta AWS SDK per PHP API Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: questo comando carica il file “local-sample.txt” in Amazon S3 e crea un oggetto con la chiave “sample.txt” nel bucket “test-files”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```


Esempio 2: questo comando carica il file “sample.txt” in Amazon S3 e crea un oggetto con la chiave “sample.txt” nel bucket “test-files”. Se il parametro -Key non viene fornito, il nome del file viene utilizzato come chiave dell'oggetto S3.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Esempio 3: questo comando carica il singolo file "local-sample.txt" su Amazon S3, creando un oggetto con la chiave prefix/to "/sample.txt" nel bucket «test-files».

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -  
File .\local-sample.txt
```

Esempio 4: questo comando carica tutti i file nella sottodirectory «Scripts» nel bucket «test-files» e applica il prefisso chiave comune "" a ciascun oggetto. SampleScripts Ogni file caricato avrà una chiave di "" SampleScripts/filename dove 'filename' varia.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\
```

Esempio 5: questo comando carica tutti i file*.ps1 nella directory locale «Scripts» nel bucket «test-files» e applica il prefisso chiave comune "" a ciascun oggetto. SampleScripts Ogni file caricato avrà una chiave "" dove 'filename' varia. SampleScripts/filename.ps1

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\ -SearchPattern *.ps1
```

Esempio 6: questo comando crea un nuovo oggetto S3 contenente la stringa di contenuto specificata con la chiave “sample.txt”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object  
contents"
```

Esempio 7: questo comando carica il file specificato (il nome del file viene utilizzato come chiave) e applica i tag specificati al nuovo oggetto.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet  
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Esempio 8: questo comando carica in modo ricorsivo la cartella specificata e applica i tag specificati a tutti i nuovi oggetti.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"
-Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Per i dettagli sull'API, vedere [PutObject](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: questo comando carica il file "local-sample.txt" in Amazon S3 e crea un oggetto con la chiave "sample.txt" nel bucket "test-files".

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

Esempio 2: questo comando carica il file "sample.txt" in Amazon S3 e crea un oggetto con la chiave "sample.txt" nel bucket "test-files". Se il parametro -Key non viene fornito, il nome del file viene utilizzato come chiave dell'oggetto S3.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Esempio 3: questo comando carica il singolo file "local-sample.txt" su Amazon S3, creando un oggetto con la chiave prefix/to "/sample.txt" nel bucket «test-files».

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -
File .\local-sample.txt
```

Esempio 4: questo comando carica tutti i file nella sottodirectory «Scripts» nel bucket «test-files» e applica il prefisso chiave comune "" a ciascun oggetto. SampleScripts Ogni file caricato avrà una chiave di "" SampleScripts/filename dove 'filename' varia.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
SampleScripts\
```

Esempio 5: questo comando carica tutti i file*.ps1 nella directory locale «Scripts» nel bucket «test-files» e applica il prefisso chiave comune "" a ciascun oggetto. SampleScripts Ogni file caricato avrà una chiave "" dove 'filename' varia. SampleScripts/filename.ps1

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\ -SearchPattern *.ps1
```

Esempio 6: questo comando crea un nuovo oggetto S3 contenente la stringa di contenuto specificata con la chiave "sample.txt".

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object  
contents"
```

Esempio 7: questo comando carica il file specificato (il nome del file viene utilizzato come chiave) e applica i tag specificati al nuovo oggetto.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet  
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Esempio 8: questo comando carica in modo ricorsivo la cartella specificata e applica i tag specificati a tutti i nuovi oggetti.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"  
-Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Per i dettagli sull'API, vedere [PutObject](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Python

SDK per Python (Boto3)

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
class ObjectWrapper:  
    """Encapsulates S3 object actions."""  
  
    def __init__(self, s3_object):  
        """
```

```

        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                                that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def put(self, data):
        """
        Upload data to the object.

        :param data: The data to upload. This can either be bytes or a string.
When this
                                argument is a string, it is interpreted as a file name,
which is
                                opened in read bytes mode.
        """
        put_data = data
        if isinstance(data, str):
            try:
                put_data = open(data, "rb")
            except IOError:
                logger.exception("Expected file name or binary data, got '%s'.",
data)
                raise

        try:
            self.object.put(Body=put_data)
            self.object.wait_until_exists()
            logger.info(
                "Put object '%s' to bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't put object '%s' to bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
            raise
        finally:
            if getattr(put_data, "close", None):

```

```
put_data.close()
```

Carica un oggetto utilizzando una richiesta condizionale.

```
class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def put_object_conditional(self, object_key: str, source_bucket: str, data:
bytes):
        """
        Uploads an object to Amazon S3 with a conditional request. Prevents
        overwrite
        using an IfNoneMatch condition for the object key.

        :param object_key: The key of the object to upload.
        :param source_bucket: The source bucket of the object.
        :param data: The data to upload.
        """
        try:
            self.s3.put_object(
                Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch="*"
            )
            print(
                f"\tConditional write successful for key {object_key} in bucket
{source_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
```

```
if error_code == "PreconditionFailed":
    print("\tConditional write failed: Precondition failed")
else:
    logger.error(f"Unexpected error: {error_code}")
    raise
```

- Per i dettagli sull'API, consulta [PutObject AWS SDK for Python \(Boto3\) API Reference](#).

Ruby

SDK per Ruby

Note

C'è di più su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Carica un file utilizzando un uploader gestito (`Object.upload_file`).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
```

```

    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
    #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
    object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__

```

Carica un file utilizzando. Object.put

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, 'rb') do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{@object.key}. Here's why:
    #{e.message}"
  end
end

```

```

    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__

```

Carica un file utilizzando `Object.put` e aggiungi la crittografia lato server.

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:

```



```
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
    object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
    #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Per i dettagli sull'API, consulta la sezione AWS SDK per Ruby API [PutObjectReference](#).

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
pub async fn upload_object(
  client: &aws_sdk_s3::Client,
  bucket_name: &str,
  file_name: &str,
  key: &str,
) -> Result<aws_sdk_s3::operation::put_object::PutObjectOutput, S3ExampleError> {
  let body =
    aws_sdk_s3::primitives::ByteStream::from_path(std::path::Path::new(file_name)).await;
  client
    .put_object()
    .bucket(bucket_name)
    .key(key)
    .body(body.unwrap())
```

```
.send()  
.await  
.map_err(S3ExampleError::from)  
}
```

- Per i dettagli sulle API, consulta la [PutObject](#) guida di riferimento all'API AWS SDK for Rust.

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
"Get contents of file from application server."  
DATA lv_body TYPE xstring.  
OPEN DATASET iv_file_name FOR INPUT IN BINARY MODE.  
READ DATASET iv_file_name INTO lv_body.  
CLOSE DATASET iv_file_name.  
  
"Upload/put an object to an S3 bucket."  
TRY.  
    lo_s3->putobject(  
        iv_bucket = iv_bucket_name  
        iv_key = iv_file_name  
        iv_body = lv_body ).  
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Per i dettagli sulle API, [PutObject](#) consulta AWS SDK for SAP ABAP API reference.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import AWSS3
import Smithy

public func uploadFile(bucket: String, key: String, file: String) async
throws {
    let fileUrl = URL(fileURLWithPath: file)
    do {
        let fileData = try Data(contentsOf: fileUrl)
        let dataStream = ByteStream.data(fileData)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )

        _ = try await client.putObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Putting an object. "))
        throw error
    }
}
```

```
import AWSS3
import Smithy

public func createFile(bucket: String, key: String, withData data: Data)
async throws {
    let dataStream = ByteStream.data(data)
```

```
let input = PutObjectInput(  
    body: dataStream,  
    bucket: bucket,  
    key: key  
)  
  
do {  
    _ = try await client.putObject(input: input)  
}  
catch {  
    print("ERROR: ", dump(error, name: "Putting an object."))  
    throw error  
}  
}
```

- Per i dettagli sull'API, consulta la [PutObject](#) guida di riferimento all'API AWS SDK for Swift.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutObjectAc1** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare PutObjectAc1.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. Puoi vedere questa azione nel contesto nel seguente esempio di codice:

- [Gestire le liste di controllo degli accessi \(ACL\)](#)

C++

SDK per C++

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
bool AwsDoc::S3::putObjectAcl(const Aws::String &bucketName, const Aws::String
&objectKey, const Aws::String &ownerID,
                                const Aws::String &granteePermission, const
    Aws::String &granteeType,
                                const Aws::String &granteeID, const Aws::String
&granteeEmailAddress,
                                const Aws::String &granteeURI, const
    Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutObjectAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);
    request.SetKey(objectKey);
```

```

    Aws::S3::Model::PutObjectAclOutcome outcome =
        s3Client.PutObjectAcl(request);

    if (!outcome.IsSuccess()) {
        auto error = outcome.GetError();
        std::cerr << "Error: putObjectAcl: " << error.GetExceptionName()
            << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the object '" << objectKey
            << "' in the bucket '" << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
    enumeration.
/*!
    \param access: Human readable string.
    \return Permission: Permission enumeration.
*/
Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
    enumeration.
/*!
    \param type: Human readable string.
    \return Type: Type enumeration.
*/
Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")

```

```

        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}

```

- Per i dettagli sull'API, [PutObjectAcl](#) consulta AWS SDK per C++ API Reference.

CLI

AWS CLI

Il comando seguente concede `full control` a due AWS utenti (`user1@example.com` e `user2@example.com`) `read` l'autorizzazione a tutti:

```

aws s3api put-object-acl --bucket amzn-s3-demo-bucket --key file.txt --grant-
full-control emailaddress=user1@example.com,emailaddress=user2@example.com --
grant-read uri=http://acs.amazonaws.com/groups/global/AllUsers

```

Vedi <http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html> i dettagli sugli ACL personalizzati (i comandi ACL `s3api`, ad esempio `put-object-acl`, usano la stessa notazione abbreviata degli argomenti).

- Per i dettagli sull'API, consulta Command Reference. [PutObjectAcl](#) AWS CLI

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

class ObjectWrapper:

```

```

"""Encapsulates S3 object actions."""

def __init__(self, s3_object):
    """
    :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                        that wraps object actions in a class-like structure.
    """
    self.object = s3_object
    self.key = self.object.key

def put_acl(self, email):
    """
    Applies an ACL to the object that grants read access to an AWS user
identified
    by email address.

    :param email: The email address of the user to grant access.
    """
    try:
        acl = self.object.Acl()
        # Putting an ACL overwrites the existing ACL, so append new grants
        # if you want to preserve existing grants.
        grants = acl.grants if acl.grants else []
        grants.append(
            {
                "Grantee": {"Type": "AmazonCustomerByEmail", "EmailAddress":
email},
                "Permission": "READ",
            }
        )
        acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
        logger.info("Granted read access to %s.", email)
    except ClientError:
        logger.exception("Couldn't add ACL to object '%s'.", self.object.key)
        raise

```

- Per i dettagli sull'API, consulta [PutObjectAcl AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.  
  " Example: Grant read access to an AWS user  
  " iv_grantread = 'emailAddress=user@example.com'  
  lo_s3->putobjectacl(  
    iv_bucket = iv_bucket_name  
    iv_key = iv_object_key  
    iv_grantread = iv_grantread ).  
  MESSAGE 'Object ACL updated.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
  MESSAGE 'Bucket does not exist.' TYPE 'E'.  
CATCH /aws1/cx_s3_nosuchkey.  
  MESSAGE 'Object key does not exist.' TYPE 'E'.  
ENDTRY.
```

- Per i dettagli sulle API, [PutObjectAcl](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutObjectLegalHold** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare PutObjectLegalHold.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Bloccare gli oggetti di Amazon S3](#)

.NET

SDK per .NET

Note

C'è altro su GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>
/// Set or modify a legal hold on an object in an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The key of the object.</param>
/// <param name="holdStatus">The On or Off status for the legal hold.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyObjectLegalHold(string bucketName,
    string objectKey, ObjectLockLegalHoldStatus holdStatus)
{
    try
    {
        var request = new PutObjectLegalHoldRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
            LegalHold = new ObjectLockLegalHold()
            {
                Status = holdStatus
            }
        };

        var response = await _amazonS3.PutObjectLegalHoldAsync(request);
        Console.WriteLine($"\\tModified legal hold for {objectKey} in
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tError modifying legal hold: '{ex.Message}'");
        return false;
    }
}
```

```
}
```

- Per i dettagli sull'API, [PutObjectLegalHold](#) consulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Come applicare lo stato relativo alla conservazione a fini legali a un oggetto

L'esempio `put-object-legal-hold` seguente imposta un blocco a fini legali sull'oggetto `doc1.rtf`.

```
aws s3api put-object-legal-hold \  
  --bucket amzn-s3-demo-bucket-with-object-lock \  
  --key doc1.rtf \  
  --legal-hold Status=ON
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [PutObjectLegalHold AWS CLI Command Reference](#).

Go

SDK per Go V2

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (  
  "bytes"  
  "context"  
  "errors"  
  "fmt"  
  "log"  
  "time"
```

```
"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
    string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
{
    input := &s3.PutObjectLegalHoldInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        LegalHold: &types.ObjectLockLegalHold{
            Status: legalHoldStatus,
        },
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }

    _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}
```

- Per i dettagli sull'API, [PutObjectLegalHold](#) consulta AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Set or modify a legal hold on an object in an S3 bucket.
public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
    ObjectLockLegalHold legalHold ;
    if (legalHoldOn) {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.ON)
            .build();
    } else {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.OFF)
            .build();
    }

    PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .legalHold(legalHold)
        .build();

    getClient().putObjectLegalHold(legalHoldRequest) ;
    System.out.println("Modified legal hold for " + objectKey + " in
"+bucketName +".");
}
```

- Per i dettagli sull'API, [PutObjectLegalHold](#) consulta AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import {
  PutObjectLegalHoldCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Apply a legal hold configuration to the specified object.
 * @param {{ bucketName: string, objectKey: string, legalHoldStatus: "ON" | "OFF" }}
 */
export const main = async ({ bucketName, objectKey, legalHoldStatus }) => {
  if (!["OFF", "ON"].includes(legalHoldStatus.toUpperCase())) {
    throw new Error(
      "Invalid parameter. legalHoldStatus must be 'ON' or 'OFF'."
    );
  }

  const client = new S3Client({});
  const command = new PutObjectLegalHoldCommand({
    Bucket: bucketName,
    Key: objectKey,
    LegalHold: {
      // Set the status to 'ON' to place a legal hold on the object.
      // Set the status to 'OFF' to remove the legal hold.
      Status: legalHoldStatus,
    },
  });

  try {
```

```
    await client.send(command);
    console.log(
      `Legal hold status set to "${legalHoldStatus}" for "${objectKey}" in
"${bucketName}"`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying legal hold status for "${objectKey}" in
"${bucketName}". The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying legal hold status for "${objectKey}" in
"${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    objectKey: {
      type: "string",
      required: true,
    },
    legalHoldStatus: {
      type: "string",
```

```

        default: "ON",
    },
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
        main(results.values);
    } else {
        console.error(errors.join("\n"));
    }
}
}

```

- Per i dettagli sull'API, [PutObjectLegalHold](#) consulta AWS SDK per JavaScript API Reference.

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Applica a un oggetto lo stato relativo alla conservazione a fini legali.

```

def set_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Set a legal hold on a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to set the legal hold on.
    """
    print()

```



```

logger.info("Setting legal hold on file [%s] in bucket [%s]", key, bucket)
try:
    before_status = "OFF"
    after_status = "ON"
    s3_client.put_object_legal_hold(
        Bucket=bucket, Key=key, LegalHold={"Status": after_status}
    )
    logger.debug(
        "Legal hold set successfully on file [%s] in bucket [%s]", key,
bucket
    )
    _print_legal_hold_update(bucket, key, before_status, after_status)
except Exception as e:
    logger.error(
        "Failed to set legal hold on file [%s] in bucket [%s]: %s", key,
bucket, e
    )

```

- Per i dettagli sull'API, consulta [PutObjectLegalHold AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

TRY.

```

" Example: Set legal hold status to ON
" iv_status = 'ON'
lo_s3->putobjectlegalhold(
    iv_bucket = iv_bucket_name
    iv_key = iv_object_key
    io_legalhold = NEW /aws1/cl_s3_objlocklegalhold(

```

```
        iv_status = iv_status ) ).  
        MESSAGE 'Object legal hold status set.' TYPE 'I'.  
    CATCH /aws1/cx_s3_nosuchbucket.  
        MESSAGE 'Bucket does not exist.' TYPE 'E'.  
    CATCH /aws1/cx_s3_nosuchkey.  
        MESSAGE 'Object key does not exist.' TYPE 'E'.  
    ENDTRY.
```

- Per i dettagli sulle API, [PutObjectLegalHold](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutObjectLockConfiguration** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare PutObjectLockConfiguration.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Bloccare gli oggetti di Amazon S3](#)

.NET

SDK per .NET

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Imposta la configurazione Object Lock di un bucket.

```
/// <summary>  
/// Enable object lock on an existing bucket.  
/// </summary>  
/// <param name="bucketName">The name of the bucket to modify.</param>
```

```

    /// <returns>True if successful.</returns>
    public async Task<bool> EnableObjectLockOnBucket(string bucketName)
    {
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
                },
            };

            var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
            Console.WriteLine($"\\tAdded an object lock policy to bucket
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
            return false;
        }
    }
}

```

Imposta il periodo di conservazione predefinito di un bucket.

```

    /// <summary>

```

```

    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
    {
        var enabledString = enableObjectLock ? "Enabled" : "Disabled";
        var timeDifference = retainUntilDate.Subtract(DateTime.Now);
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                    Rule = new ObjectLockRule()
                    {
                        DefaultRetention = new DefaultRetention()
                        {
                            Mode = retention,
                            Days = timeDifference.Days // Can be specified in
days or years but not both.
                        }
                    }
                }
            };

```

```
        var response = await
        _amazonS3.PutObjectLockConfigurationAsync(request);
        Console.WriteLine($"\\tAdded a default retention to bucket
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tError modifying object lock: '{ex.Message}'");
        return false;
    }
}
```

- Per i dettagli sull'API, [PutObjectLockConfiguration](#) consulta AWS SDK per .NET API Reference.

CLI

AWS CLI

Come impostare la configurazione Object Lock di un bucket

L'esempio `put-object-lock-configuration` seguente imposta un Object Lock di 50 giorni sul bucket specificato.

```
aws s3api put-object-lock-configuration \
    --bucket amzn-s3-demo-bucket-with-object-lock \
    --object-lock-configuration '{ "ObjectLockEnabled": "Enabled", "Rule":
    { "DefaultRetention": { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [PutObjectLockConfiguration AWS CLI](#) Command Reference.

Go

SDK per Go V2

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Imposta la configurazione Object Lock di un bucket.

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
    string) error {
    // Versioning must be enabled on the bucket before object locking is enabled.
    verInput := &s3.PutBucketVersioningInput{
        Bucket: aws.String(bucket),
        VersioningConfiguration: &types.VersioningConfiguration{
```

```

    MFADelete: types.MFADeleteDisabled,
    Status:    types.BucketVersioningStatusEnabled,
  },
}
_, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
    return err
}

input := &s3.PutObjectLockConfigurationInput{
    Bucket: aws.String(bucket),
    ObjectLockConfiguration: &types.ObjectLockConfiguration{
        ObjectLockEnabled: types.ObjectLockEnabledEnabled,
    },
}
_, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
}

return err
}

```

Imposta il periodo di conservazione predefinito di un bucket.

```

import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"

```

```
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// ModifyDefaultBucketRetention modifies the default retention period of an
// existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
    ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
    retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: lockMode,
            Rule: &types.ObjectLockRule{
                DefaultRetention: &types.DefaultRetention{
                    Days: aws.Int32(retentionPeriod),
                    Mode: retentionMode,
                },
            },
        },
    }

    _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    }
}
```



```
    return err
}
```

- Per i dettagli sull'API, consulta la [PutObjectLockConfiguration](#) sezione AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Imposta la configurazione Object Lock di un bucket.

```
// Enable object lock on an existing bucket.
public void enableObjectLockOnBucket(String bucketName) {
    try {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        // Enable versioning on the bucket.
        getClient().putBucketVersioning(putBucketVersioningRequest);
        PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(ObjectLockConfiguration.builder()
                .objectLockEnabled(ObjectLockEnabled.ENABLED)
                .build())
    }
```

```
        .build();

        getClient().putObjectLockConfiguration(request);
        System.out.println("Successfully enabled object lock on
"+bucketName);

    } catch (S3Exception ex) {
        System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
    }
}
```

Imposta il periodo di conservazione predefinito di un bucket.

```
// Set or modify a retention period on an S3 bucket.
public void modifyBucketDefaultRetention(String bucketName) {
    VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
        .mfaDelete(MFADelete.DISABLED)
        .status(BucketVersioningStatus.ENABLED)
        .build();

    PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

    getClient().putBucketVersioning(versioningRequest);
    DefaultRetention rention = DefaultRetention.builder()
        .days(1)
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .build();

    ObjectLockRule lockRule = ObjectLockRule.builder()
        .defaultRetention(rention)
        .build();

    ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
        .objectLockEnabled(ObjectLockEnabled.ENABLED)
        .rule(lockRule)
```

```
        .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .objectLockConfiguration(objectLockConfiguration)
        .build();

getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName
+ ".");
    }
```

- Per i dettagli sull'API, consulta la [PutObjectLockConfiguration](#) sezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Imposta la configurazione Object Lock di un bucket.

```
import {
    PutObjectLockConfigurationCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Enable S3 Object Lock for an Amazon S3 bucket.
 * After you enable Object Lock on a bucket, you can't
 * disable Object Lock or suspend versioning for that bucket.
 * @param {{ bucketName: string, enabled: boolean }}
 */
```

```
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new PutObjectLockConfigurationCommand({
    Bucket: bucketName,
    // The Object Lock configuration that you want to apply to the specified
    bucket.
    ObjectLockConfiguration: {
      ObjectLockEnabled: "Enabled",
    },
  });

  try {
    await client.send(command);
    console.log(`Object Lock for "${bucketName}" enabled.`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying the object lock configuration for the
        bucket "${bucketName}". The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying the object lock configuration for the
        bucket "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
```

```

        type: "string",
        required: true,
    },
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
        main(results.values);
    } else {
        console.error(errors.join("\n"));
    }
}

```

Imposta il periodo di conservazione predefinito di un bucket.

```

import {
    PutObjectLockConfigurationCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Change the default retention settings for an object in an Amazon S3 bucket.
 * @param {{ bucketName: string, retentionDays: string }}
 */
export const main = async ({ bucketName, retentionDays }) => {
    const client = new S3Client({});

    try {
        await client.send(
            new PutObjectLockConfigurationCommand({
                Bucket: bucketName,
                // The Object Lock configuration that you want to apply to the specified
                bucket.
                ObjectLockConfiguration: {
                    ObjectLockEnabled: "Enabled",

```

```

        Rule: {
            // The default Object Lock retention mode and period that you want to
            apply
            // to new objects placed in the specified bucket. Bucket settings
            require
            // both a mode and a period. The period can be either Days or Years
            but
            // you must select one.
            DefaultRetention: {
                // In governance mode, users can't overwrite or delete an object
                version
                // or alter its lock settings unless they have special permissions.
                With
                // governance mode, you protect objects against being deleted by
                most users,
                // but you can still grant some users permission to alter the
                retention settings
                // or delete the objects if necessary.
                Mode: "GOVERNANCE",
                Days: Number.parseInt(retentionDays),
            },
        },
    },
    }),
);
console.log(
    `Set default retention mode to "GOVERNANCE" with a retention period of
    ${retentionDays} day(s).`,
);
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while setting the default object retention for a bucket.
            The bucket doesn't exist.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while setting the default object retention for a bucket.
            ${caught.name}: ${caught.message}`,
        );
    } else {

```

```
        throw caught;
    }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
    isMain,
    validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
    const options = {
        bucketName: {
            type: "string",
            required: true,
        },
        retentionDays: {
            type: "string",
            required: true,
        },
    };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
        main(results.values);
    } else {
        console.error(errors.join("\n"));
    }
}
```

- Per i dettagli sull'API, consulta la [PutObjectLockConfiguration](#) sezione AWS SDK per JavaScript API Reference.

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Abilita la configurazione Object Lock.

```
s3_client.put_object_lock_configuration(  
    Bucket=bucket,  
    ObjectLockConfiguration={"ObjectLockEnabled": "Disabled", "Rule":  
{}},  
)
```

- Per i dettagli sull'API, consulta [PutObjectLockConfiguration AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.  
    " Example: Enable object lock with default retention  
    " iv_enabled = 'Enabled'  
    lo_s3->putobjectlockconfiguration(  
        iv_bucket = iv_bucket_name  
        io_objectlockconfiguration = NEW /aws1/cl_s3_objectlockconf(  
            iv_objectlockenabled = iv_enabled ) ).  
    MESSAGE 'Object lock configuration set.' TYPE 'I'.
```



```
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Per i dettagli sulle API, [PutObjectLockConfiguration](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutObjectRetention** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare PutObjectRetention.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Bloccare gli oggetti di Amazon S3](#)

.NET

SDK per .NET

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/// <summary>
/// Set or modify a retention period on an object in an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The key of the object.</param>
/// <param name="retention">The retention mode.</param>
/// <param name="retainUntilDate">The date retention expires.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
```

```

        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }
}

```

- Per i dettagli sull'API, consulta la [PutObjectRetention](#) sezione AWS SDK per .NET API Reference.

CLI

AWS CLI

Come impostare la configurazione di conservazione degli oggetti per un oggetto

L'esempio `put-object-retention` seguente imposta una configurazione di conservazione degli oggetti per l'oggetto specificato fino al giorno 01/01/2025.

```
aws s3api put-object-retention \
```

```
--bucket amzn-s3-demo-bucket-with-object-lock \  
--key doc1.rtf \  
--retention '{ "Mode": "GOVERNANCE", "RetainUntilDate":  
"2025-01-01T00:00:00" }'
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [PutObjectRetention AWS CLI Command Reference](#).

Go

SDK per Go V2

 Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}
```

```
// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
error {
    input := &s3.PutObjectRetentionInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        Retention: &types.ObjectLockRetention{
            Mode:          retentionMode,
            RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
        },
        BypassGovernanceRetention: aws.Bool(true),
    }

    _, err := actor.S3Client.PutObjectRetention(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}
```

- Per i dettagli sull'API, consulta la [PutObjectRetention](#) sezione AWS SDK per Go API Reference.

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
// Set or modify a retention period on an object in an S3 bucket.
public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
{
    // Calculate the instant one day from now.
    Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

    // Convert the Instant to a ZonedDateTime object with a specific time
    zone.
    ZonedDateTime zonedDateTime =
futureInstant.atZone(ZoneId.systemDefault());

    // Define a formatter for human-readable output.
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

    // Format the ZonedDateTime object to a human-readable date string.
    String humanReadableDate = formatter.format(zonedDateTime);

    // Print the formatted date string.
    System.out.println("Formatted Date: " + humanReadableDate);
    ObjectLockRetention retention = ObjectLockRetention.builder()
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .retainUntilDate(futureInstant)
        .build();

    PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .retention(retention)
        .build();

    getClient().putObjectRetention(retentionRequest);
    System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
}
```

- Per i dettagli sull'API, consulta la [PutObjectRetention](#) sezione AWS SDK for Java 2.x API Reference.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import {
  PutObjectRetentionCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Place a 24-hour retention period on an object in an Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const command = new PutObjectRetentionCommand({
    Bucket: bucketName,
    Key: key,
    BypassGovernanceRetention: false,
    Retention: {
      // In governance mode, users can't overwrite or delete an object version
      // or alter its lock settings unless they have special permissions. With
      // governance mode, you protect objects against being deleted by most
      users,
      // but you can still grant some users permission to alter the retention
      settings
      // or delete the objects if necessary.
      Mode: "GOVERNANCE",
      RetainUntilDate: new Date(new Date().getTime() + 24 * 60 * 60 * 1000),
    },
  });

  try {
    await client.send(command);
    console.log("Object Retention settings updated.");
  }
```

```
    } catch (caught) {
      if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
      ) {
        console.error(
          `Error from S3 while modifying the governance mode and retention period
on an object. The bucket doesn't exist.`,
        );
      } else if (caught instanceof S3ServiceException) {
        console.error(
          `Error from S3 while modifying the governance mode and retention period
on an object. ${caught.name}: ${caught.message}`,
        );
      } else {
        throw caught;
      }
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
```

```
const { errors, results } = loadArgs();
if (!errors) {
  main(results.values);
} else {
  console.error(errors.join("\n"));
}
}
```

- Per i dettagli sull'API, consulta la [PutObjectRetention](#) sezione AWS SDK per JavaScript API Reference.

PowerShell

Strumenti per PowerShell V4

Esempio 1: il comando abilita la modalità di conservazione della governance fino alla data “31 dicembre 2019 00:00:00” per l’oggetto “testfile.txt” nel bucket S3 specificato.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Per i dettagli sull'API, vedere [PutObjectRetention](#) in AWS Strumenti per PowerShell Cmdlet Reference (V4).

Strumenti per V5 PowerShell

Esempio 1: il comando abilita la modalità di conservazione della governance fino alla data “31 dicembre 2019 00:00:00” per l’oggetto “testfile.txt” nel bucket S3 specificato.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Per i dettagli sull'API, vedere [PutObjectRetention](#) in AWS Strumenti per PowerShell Cmdlet Reference (V5).

Python

SDK per Python (Boto3)

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Abilita la conservazione degli oggetti.

```
s3_client.put_object_retention(  
    Bucket=bucket,  
    Key=key,  
    VersionId=version_id,  
    Retention={"Mode": "GOVERNANCE", "RetainUntilDate":  
far_future_date},  
    BypassGovernanceRetention=True,  
)
```

- Per i dettagli sull'API, consulta [PutObjectRetention AWSSDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
TRY.  
    " Example: Set retention mode to GOVERNANCE for 30 days  
    " iv_mode = 'GOVERNANCE'  
    " iv_retain_date should be a timestamp in the future  
    lo_s3->putobjectretention(  

```

```

        iv_bucket = iv_bucket_name
        iv_key = iv_object_key
        io_retention = NEW /aws1/cl_s3_objectlockret(
            iv_mode = iv_mode
            iv_retainuntildate = iv_retain_date )
        iv_bypassgovernanceretention = abap_true ).
    MESSAGE 'Object retention set.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

```

- Per i dettagli sulle API, [PutObjectRetention](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **RestoreObject** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare RestoreObject.

.NET

SDK per .NET

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>

```

```

/// This example shows how to restore an archived object in an Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class RestoreArchivedObject
{
    public static void Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string objectKey = "archived-object.txt";

        // Specify your bucket region (an example region is shown).
        RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

        IAmazonS3 client = new AmazonS3Client(bucketRegion);
        RestoreObjectAsync(client, bucketName, objectKey).Wait();
    }

    /// <summary>
    /// This method restores an archived object from an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// RestoreObjectAsync.</param>
    /// <param name="bucketName">A string representing the name of the
    /// bucket where the object was located before it was archived.</param>
    /// <param name="objectKey">A string representing the name of the
    /// archived object to restore.</param>
    public static async Task RestoreObjectAsync(IAmazonS3 client, string
bucketName, string objectKey)
    {
        try
        {
            var restoreRequest = new RestoreObjectRequest
            {
                BucketName = bucketName,
                Key = objectKey,
                Days = 2,
            };
            RestoreObjectResponse response = await
client.RestoreObjectAsync(restoreRequest);

            // Check the status of the restoration.
            await CheckRestorationStatusAsync(client, bucketName, objectKey);
        }
    }
}

```

```

        catch (AmazonS3Exception amazonS3Exception)
        {
            Console.WriteLine($"Error: {amazonS3Exception.Message}");
        }
    }

    /// <summary>
    /// This method retrieves the status of the object's restoration.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// GetObjectMetadataAsync.</param>
    /// <param name="bucketName">A string representing the name of the Amazon
    /// S3 bucket which contains the archived object.</param>
    /// <param name="objectKey">A string representing the name of the
    /// archived object you want to restore.</param>
    public static async Task CheckRestorationStatusAsync(IAmazonS3 client,
string bucketName, string objectKey)
    {
        GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
        };

        GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);

        var restStatus = response.RestoreInProgress ? "in-progress" :
"finished or failed";
        Console.WriteLine($"Restoration status: {restStatus}");
    }
}

```

- Per i dettagli sull'API, consulta la [RestoreObject](#) sezione AWS SDK per .NET API Reference.

CLI

AWS CLI

Come creare una richiesta di ripristino per un oggetto

L'esempio `restore-object` seguente ripristina l'oggetto Amazon S3 Glacier specificato per il bucket `my-glacier-bucket` per 10 giorni.

```
aws s3api restore-object \  
  --bucket my-glacier-bucket \  
  --key doc1.rtf \  
  --restore-request Days=10
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [RestoreObject AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.RestoreRequest;  
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;  
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import software.amazon.awssdk.services.s3.model.Tier;  
  
/*  
 * For more information about restoring an object, see "Restoring an archived  
 * object" at  
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html  
 */
```

```

* Before running this Java V2 code example, set up your development
environment, including your credentials.
*
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <expectedBucketOwner>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name of an object with a Storage class value of
Glacier.\s
                expectedBucketOwner - The account that owns the bucket (you can
obtain this value from the AWS Management Console).\s
            "";

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String expectedBucketOwner = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
        s3.close();
    }

    /**
     * Restores an S3 object from the Glacier storage class.
     *

```

```

    * @param s3                an instance of the {@link S3Client} to be used
    for interacting with Amazon S3
    * @param bucketName        the name of the S3 bucket where the object is
    stored
    * @param keyName            the key (object name) of the S3 object to be
    restored
    * @param expectedBucketOwner the AWS account ID of the expected bucket
    owner
    */
    public static void restoreS3Object(S3Client s3, String bucketName, String
    keyName, String expectedBucketOwner) {
        try {
            RestoreRequest restoreRequest = RestoreRequest.builder()
                .days(10)

            .glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
                .build();

            RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
                .expectedBucketOwner(expectedBucketOwner)
                .bucket(bucketName)
                .key(keyName)
                .restoreRequest(restoreRequest)
                .build();

            s3.restoreObject(objectRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}

```

- Per i dettagli sull'API, consulta la [RestoreObject](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **SelectObjectContent** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare **SelectObjectContent**.

CLI

AWS CLI

Come filtrare il contenuto di un oggetto Amazon S3 in base a un'istruzione SQL

L'esempio `select-object-content` seguente filtra l'oggetto `my-data-file.csv` con l'istruzione SQL specificata e invia l'output a un file.

```
aws s3api select-object-content \
  --bucket amzn-s3-demo-bucket \
  --key my-data-file.csv \
  --expression "select * from s3object limit 100" \
  --expression-type 'SQL' \
  --input-serialization '{"CSV": {}, "CompressionType": "NONE"}' \
  --output-serialization '{"CSV": {}}' "output.csv"
```

Questo comando non produce alcun output.

- Per i dettagli sull'API, consulta [SelectObjectContent AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Il seguente esempio mostra una query mediante un oggetto JSON. L'[esempio completo](#) mostra anche l'uso di un oggetto CSV.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
```



```
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import
    software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

public class SelectObjectContentExample {
    static final Logger logger =
        LoggerFactory.getLogger(SelectObjectContentExample.class);
    static String BUCKET_NAME = "amzn-s3-demo-bucket"; // Replace with your
        bucket name.
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
        master/dist/countries.csv";
    static String URL_JSON = "https://raw.githubusercontent.com/mledoze/
        countries/master/dist/countries.json";

    public static void main(String[] args) {
        SelectObjectContentExample selectObjectContentExample = new
            SelectObjectContentExample();
    }
}
```

```

    try {
        SelectObjectContentExample.setUp();
        selectObjectContentExample.runSelectObjectContentMethodForJSON();
        selectObjectContentExample.runSelectObjectContentMethodForCSV();
    } catch (SdkException e) {
        logger.error(e.getMessage(), e);
        System.exit(1);
    } finally {
        SelectObjectContentExample.tearDown();
    }
}

EventStreamInfo runSelectObjectContentMethodForJSON() {
    // Set up request parameters.
    final String queryExpression = "select * from s3object[*][*] c where
c.area < 350000";
    final String fileType = FILE_JSON;

    InputSerialization inputSerialization = InputSerialization.builder()
        .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
        .compressionType(CompressionType.NONE)
        .build();

    OutputSerialization outputSerialization = OutputSerialization.builder()
        .json(JSONOutput.builder().recordDelimiter(null).build())
        .build();

    // Build the SelectObjectContentRequest.
    SelectObjectContentRequest select = SelectObjectContentRequest.builder()
        .bucket(BUCKET_NAME)
        .key(FILE_JSON)
        .expression(queryExpression)
        .expressionType(ExpressionType.SQL)
        .inputSerialization(inputSerialization)
        .outputSerialization(outputSerialization)
        .build();

    EventStreamInfo eventStreamInfo = new EventStreamInfo();
    // Call the selectObjectContent method with the request and a response
    handler.
    // Supply an EventStreamInfo object to the response handler to gather
    records and information from the response.
    s3AsyncClient.selectObjectContent(select,
    buildResponseHandler(eventStreamInfo)).join();
}

```

```

        // Log out information gathered while processing the response stream.
        long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record
->
            record.split("\n").length
        ).sum();
        logger.info("Total records {}: {}", fileType, recordCount);
        logger.info("Visitor onRecords for fileType {} called {} times",
fileType, eventStreamInfo.getCountOnRecordsCalled());
        logger.info("Visitor onStats for fileType {}, {}", fileType,
eventStreamInfo.getStats());
        logger.info("Visitor onContinuations for fileType {}, {}", fileType,
eventStreamInfo.getCountContinuationEvents());
        return eventStreamInfo;
    }

    static SelectObjectContentResponseHandler
buildResponseHandler(EventStreamInfo eventStreamInfo) {
    // Use a Visitor to process the response stream. This visitor logs
information and gathers details while processing.
    final SelectObjectContentResponseHandler.Visitor visitor =
SelectObjectContentResponseHandler.Visitor.builder()
        .onRecords(r -> {
            logger.info("Record event received.");
            eventStreamInfo.addRecord(r.payload().asUtf8String());
            eventStreamInfo.incrementOnRecordsCalled();
        })
        .onCont(ce -> {
            logger.info("Continuation event received.");
            eventStreamInfo.incrementContinuationEvents();
        })
        .onProgress(pe -> {
            Progress progress = pe.details();
            logger.info("Progress event received:\n bytesScanned:
{}\nbytesProcessed: {}\nbytesReturned:{}",
                progress.bytesScanned(),
                progress.bytesProcessed(),
                progress.bytesReturned());
        })
        .onEnd(ee -> logger.info("End event received.))
        .onStats(se -> {
            logger.info("Stats event received.");
            eventStreamInfo.addStats(se.details());
        })
    }

```

```
        .build();

        // Build the SelectObjectContentResponseHandler with the visitor that
        // processes the stream.
        return SelectObjectContentResponseHandler.builder()
            .subscriber(visitor).build();
    }

    // The EventStreamInfo class is used to store information gathered while
    // processing the response stream.
    static class EventStreamInfo {
        private final List<String> records = new ArrayList<>();
        private Integer countOnRecordsCalled = 0;
        private Integer countContinuationEvents = 0;
        private Stats stats;

        void incrementOnRecordsCalled() {
            countOnRecordsCalled++;
        }

        void incrementContinuationEvents() {
            countContinuationEvents++;
        }

        void addRecord(String record) {
            records.add(record);
        }

        void addStats(Stats stats) {
            this.stats = stats;
        }

        public List<String> getRecords() {
            return records;
        }

        public Integer getCountOnRecordsCalled() {
            return countOnRecordsCalled;
        }

        public Integer getCountContinuationEvents() {
            return countContinuationEvents;
        }
    }
}
```

```
    public Stats getStats() {  
        return stats;  
    }  
}
```

- Per i dettagli sull'API, consulta la [SelectObjectContent](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **UploadPart** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare UploadPart.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- [Utilizzo dei checksum](#)
- [Utilizzare la funzionalità per l'integrità degli oggetti di Amazon S3](#)

C++

SDK per C++

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
//! Upload a part to an S3 bucket.  
/*!  
    \param bucket: The name of the S3 bucket where the object will be uploaded.  
    \param key: The unique identifier (key) for the object within the S3 bucket.  
    \param uploadID: An upload ID string.  
    \param partNumber:  
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
```

```

    \param calculatedHash: A data integrity hash to set, depending on the
    checksum algorithm,
                                ignored when it is an empty string.
    \param body: An shared_ptr IOStream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String
    &bucket,
                                                         const Aws::String &key,
                                                         const Aws::String
    &uploadID,
                                                         int partNumber,
    Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                         const Aws::String
    &calculatedHash,
                                                         const
    std::shared_ptr<Aws::IOStream> &body,
                                                         const Aws::S3::S3Client
    &client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:

```

```

        request.SetChecksumSHA1(calculatedHash);
        break;
    case Aws::S3::Model::ChecksumAlgorithm::SHA256:
        request.SetChecksumSHA256(calculatedHash);
        break;
    }
}

return client.UploadPart(request);
}

```

- Per i dettagli sull'API, consulta la [UploadPart](#) sezione AWS SDK per C++ API Reference.

CLI

AWS CLI

Il comando seguente carica la prima parte di un caricamento in più parti avviato con il comando `create-multipart-upload`.

```
aws s3api upload-part --bucket amzn-s3-demo-bucket --key
'multipart/01' --part-number 1 --body part01 --upload-id
'dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZLjF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P'
```

L'opzione `body` richiede il nome o il percorso di un file locale per il caricamento (non utilizzare il prefisso `file://`). La dimensione minima della parte è 5 MB. L'ID di caricamento viene restituito da `create-multipart-upload` e può essere recuperato anche con `list-multipart-uploads`. Il bucket e la chiave vengono specificati quando si crea il caricamento in più parti.

Output:

```
{
  "ETag": "\"e868e0f4719e394144ef36531ee6824c\""
}
```

Salva il valore `ETag` di ogni parte per utilizzarlo in un secondo momento. Tale valore sarà necessario per completare un caricamento in più parti.

- Per i dettagli sull'API, consulta [UploadPart AWS CLI Command Reference](#).

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
    let upload_part_res = client
        .upload_part()
        .key(&key)
        .bucket(&bucket_name)
        .upload_id(upload_id)
        .body(stream)
        .part_number(part_number)
        .send()
        .await?;

    upload_parts.push(
        CompletedPart::builder()
            .e_tag(upload_part_res.e_tag.unwrap_or_default())
            .part_number(part_number)
```



```
        .build(),
    );
}
```

```
// Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
// upload the file.
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await?;

let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
    "Missing upload_id after CreateMultipartUpload",
))?;
```

```
// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;
```

- Per i dettagli sulle API, consulta la [UploadPart](#) guida di riferimento all'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **UploadPartCopy** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare UploadPartCopy.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Eseguire una copia in più parti](#)

CLI

AWS CLI

Come caricare parte di un oggetto copiando i dati da un oggetto esistente come origine dati

L'esempio upload-part-copy seguente carica una parte copiando i dati da un oggetto esistente come origine dati.

```
aws s3api upload-part-copy \  
  --bucket amzn-s3-demo-bucket \  
  --key "Map_Data_June.mp4" \  
  --copy-source "amzn-s3-demo-bucket/copy_of_Map_Data_June.mp4" \  
  --part-number 1 \  
  --upload-  
id "bq0tdE1CDpWQYRPLHuNG50xAT6pA5D.m_RiBy0gg0H6b13pVRY7QjvL1f75iFdJqp_2wztk5hvpUM2SesXgrz"
```

Output:

```
{  
  "CopyPartResult": {  
    "LastModified": "2019-12-13T23:16:03.000Z",  
    "ETag": "\"711470fc377698c393d94aed6305e245\""  
  }  
}
```

- Per i dettagli sull'API, consulta [UploadPartCopy AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
        .bucket(toBucket)
        .key(key)
        .build();

    getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
        .thenApply(createMultipartUploadResponse -> {
            String uploadId = createMultipartUploadResponse.uploadId();
            System.out.println("Upload ID: " + uploadId);

            UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                .sourceBucket(bucketName)
                .destinationBucket(toBucket)
                .sourceKey(key)
                .destinationKey(key)
                .uploadId(uploadId) // Use the valid uploadId.
                .partNumber(1) // Ensure the part number is correct.
                .copySourceRange("bytes=0-1023") // Adjust range as needed
                .build();

            return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
        })
        .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
        .whenComplete((uploadPartCopyResponse, exception) -> {
            if (exception != null) {
                // Handle any exceptions.
                logger.error("Error during upload part copy: " +
exception.getMessage());
            }
        });
}
```

```
        } else {  
            // Successfully completed the upload part copy.  
            System.out.println("Upload Part Copy completed successfully.  
ETag: " + uploadPartCopyResponse.copyPartResult().eTag());  
        }  
    });  
    return null;  
}
```

- Per i dettagli sull'API, consulta la [UploadPartCopy](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Scenari per l'utilizzo di Amazon S3 AWS SDK

I seguenti esempi di codice mostrano come implementare scenari comuni in Amazon S3 con AWS SDK. Questi scenari illustrano come eseguire attività specifiche chiamando più funzioni all'interno di Amazon S3 o in combinazione con altri Servizi AWS. Ogni scenario include un collegamento al codice sorgente completo, dove è possibile trovare le istruzioni su come configurare ed eseguire il codice.

Gli scenari sono relativi a un livello intermedio di esperienza per aiutarti a comprendere le azioni di servizio nel contesto.

Esempi

- [Controllare se un bucket esiste](#)
- [Controlla se esiste un oggetto](#)
- [Convertire il testo in voce e viceversa utilizzando un AWS SDK](#)
- [Crea un URL predefinito per Amazon S3 utilizzando un AWS SDK](#)
- [Creazione di un'applicazione di gestione delle risorse fotografiche che consente agli utenti di gestire le foto utilizzando etichette](#)
- [Una pagina Web che elenca gli oggetti Amazon S3 utilizzando un AWS SDK](#)
- [Creazione di un'applicazione Amazon Textract explorer](#)
- [Elimina tutti gli oggetti in un determinato bucket Amazon S3 utilizzando un AWS SDK](#)

- [Eliminare caricamenti multiparte incompleti su Amazon S3 utilizzando un AWS SDK](#)
- [Rileva i DPI nelle immagini con Amazon Rekognition utilizzando un AWS SDK](#)
- [Rileva le entità nel testo estratto da un'immagine utilizzando un AWS SDK](#)
- [Rileva i volti in un'immagine utilizzando un AWS SDK](#)
- [Rileva oggetti nelle immagini con Amazon Rekognition utilizzando un AWS SDK](#)
- [Rileva persone e oggetti in un video con Amazon Rekognition utilizzando un AWS SDK](#)
- [Scaricare l'oggetto S3 "directory" da un bucket Amazon Simple Storage Service \(Amazon S3\)](#)
- [Scaricare tutti gli oggetti da un bucket Amazon Simple Storage Service \(Amazon S3\) in una directory locale](#)
- [Scarica uno stream di dimensioni sconosciute da un oggetto Amazon S3 utilizzando un AWS SDK](#)
- [Ottieni un oggetto Amazon S3 da un punto di Multi-Region accesso utilizzando un AWS SDK](#)
- [Ottieni un oggetto da un bucket Amazon S3 utilizzando un AWS SDK, specificando un'intestazione If-Modified-Since](#)
- [Inizia a utilizzare la crittografia per oggetti Amazon S3 utilizzando un AWS SDK](#)
- [Inizia a usare i tag per gli oggetti Amazon S3 utilizzando un AWS SDK](#)
- [Guida introduttiva ai cluster di elaborazione dei Big Data](#)
- [Guida introduttiva alla gestione della configurazione](#)
- [Guida introduttiva all'estrazione del testo dei documenti](#)
- [Guida introduttiva agli store di funzionalità di machine learning](#)
- [Guida introduttiva allo storage di oggetti](#)
- [Guida introduttiva all'analisi delle query](#)
- [Lavora con le funzionalità di blocco degli oggetti di Amazon S3 utilizzando un AWS SDK](#)
- [Effettua richieste condizionali per Amazon S3 utilizzando un AWS SDK](#)
- [Gestisci gli elenchi di controllo degli accessi \(ACL\) per i bucket Amazon S3 utilizzando un AWS SDK](#)
- [Gestisci messaggi Amazon SQS di grandi dimensioni utilizzando Amazon S3 con un AWS SDK](#)
- [Gestisci oggetti Amazon S3 con versioni in batch con una funzione Lambda utilizzando un AWS SDK](#)
- [Analizza gli URI di Amazon S3 utilizzando un AWS SDK](#)
- [Esegui una copia multiparte di un oggetto Amazon S3 utilizzando un AWS SDK](#)
- [Ricevi ed elabora le notifiche degli eventi di Amazon S3 utilizzando un AWS SDK](#)

- [Salvate EXIF e altre informazioni sull'immagine utilizzando un AWS SDK](#)
- [Invia notifiche di eventi S3 ad Amazon EventBridge utilizzando un AWS SDK](#)
- [Tieni traccia del caricamento o del download di un oggetto Amazon S3 utilizzando un AWS SDK](#)
- [Trasformare i dati per l'applicazione con S3 Object Lambda](#)
- [Esempi di approcci per test unitari e di integrazione con un AWS SDK](#)
- [Caricare in modo ricorsivo una directory locale in un bucket Amazon Simple Storage Service \(Amazon S3\)](#)
- [Carica o scarica file di grandi dimensioni da e verso Amazon S3 utilizzando un AWS SDK](#)
- [Carica un flusso di dimensioni sconosciute su un oggetto Amazon S3 utilizzando un AWS SDK](#)
- [Usa i checksum per lavorare con un oggetto Amazon S3 utilizzando un AWS SDK](#)
- [Lavora con le funzionalità di integrità degli oggetti di Amazon S3 utilizzando un AWS SDK](#)
- [Lavora con oggetti con versione di Amazon S3 utilizzando un AWS SDK](#)

Controllare se un bucket esiste

L'esempio di codice seguente mostra come verificare se un bucket esiste.

Java

SDK per Java 2.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Puoi utilizzare il seguente `doesBucketExists` metodo in sostituzione del metodo SDK for Java [V1 AmazonS3Client BucketExist #does V2](#) (String).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.http.HttpStatusCode;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.utils.Validate;
```

```

public class DoesBucketExist {
    private static final Logger logger =
        LoggerFactory.getLogger(DoesBucketExist.class);

    public static void main(String[] args) {
        DoesBucketExist doesBucketExist = new DoesBucketExist();

        final S3Client s3SyncClient = S3Client.builder().build();
        final String bucketName = "amzn-s3-demo-bucket"; // Replace with the
        bucket name that you want to check.

        boolean exists = doesBucketExist.doesBucketExist(bucketName,
        s3SyncClient);
        logger.info("Bucket exists: {}", exists);
    }

    /**
     * Checks if the specified bucket exists. Amazon S3 buckets are named in a
     * global namespace; use this method to
     * determine if a specified bucket name already exists, and therefore can't
     * be used to create a new bucket.
     * <p>
     * Internally this method uses the <a
     * href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
     services/s3/
     S3Client.html#getBucketAcl(java.util.function.Consumer)">S3Client.getBucketAcl(String)</
     a>
     * operation to determine whether the bucket exists.
     * <p>
     * This method is equivalent to the AWS SDK for Java V1's <a
     * href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/
     com/amazonaws/services/s3/AmazonS3Client.html#doesBucketExistV2-
     java.lang.String-">AmazonS3Client#doesBucketExistV2(String)</a>.
     *
     * @param bucketName The name of the bucket to check.
     * @param s3SyncClient An <code>S3Client</code> instance. The method checks
     for the bucket in the AWS Region
     * configured on the instance.
     * @return The value true if the specified bucket exists in Amazon S3; the
     value false if there is no bucket in
     * Amazon S3 with that name.
     */
    public boolean doesBucketExist(String bucketName, S3Client s3SyncClient) {

```

```
    try {
        Validate.notEmpty(bucketName, "The bucket name must not be null or an
empty string.", "");
        s3SyncClient.getBucketAcl(r -> r.bucket(bucketName));
        return true;
    } catch (AwsServiceException ase) {
        // A redirect error or an AccessDenied exception means the bucket
exists but it's not in this region
        // or we don't have permissions to it.
        if ((ase.statusCode() == HttpStatusCode.MOVED_PERMANENTLY) ||
"AccessDenied".equals(ase.awsErrorDetails().errorCode())) {
            return true;
        }
        if (ase.statusCode() == HttpStatusCode.NOT_FOUND) {
            return false;
        }
        throw ase;
    }
}
```

- Per i dettagli sull'API, consulta la sezione API Reference. [GetBucketAcl](#) AWS SDK for Java 2.x

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Controlla se esiste un oggetto

Il seguente esempio di codice mostra come verificare l'esistenza di un oggetto.

Java

SDK per Java 2.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Puoi utilizzare il seguente `doesObjectExist` metodo in sostituzione del metodo SDK for Java [V1 AmazonS3Client ObjectExist #does](#) (String, String).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.NoSuchKeyException;
import software.amazon.awssdk.utils.Validate;

public class DoesObjectExist {
    private static final Logger logger =
        LoggerFactory.getLogger(DoesObjectExist.class);

    public static void main(String[] args) {
        DoesObjectExist doesObjectExist = new DoesObjectExist();

        final S3Client s3SyncClient = S3Client.builder().build();
        final String bucketName = "amzn-s3-demo-bucket"; // Replace with your
        bucket name.
        final String key = "my-key"; // Replace with your object key.

        boolean exists = doesObjectExist.doesObjectExist(bucketName, key,
        s3SyncClient);
        logger.info("Object exists: {}", exists);
    }

    /**
     * Checks if the specified object exists in the specified bucket.
     * <p>
     * Internally this method uses the
     * <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
     services/s3/
     S3Client.html#headObject(java.util.function.Consumer)">S3Client.headObject</a>
     * operation to determine whether the object exists.
     * <p>
     * This method is equivalent to the AWS SDK for Java V1's
     * <a href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/
     amazonaws/services/s3/AmazonS3Client.html#doesObjectExist-java.lang.String-
     java.lang.String-">AmazonS3Client#doesObjectExist(String, String)</a>.
     * <p>
```

```

    * <b>Note:</b> This method returns {@code false} only when S3 responds with
    404 (NoSuchKey). If the caller
    * does not have {@code s3:ListBucket} permission on the bucket, S3 may
    return 403 instead of 404 for
    * non-existent objects, which will be thrown as an exception.
    *
    * @param bucketName The name of the bucket containing the object.
    * @param key         The key of the object to check.
    * @param s3SyncClient An {@code S3Client} instance.
    * @return {@code true} if the object exists; {@code false} if it does not
    exist.
    */
    public boolean doesObjectExist(String bucketName, String key, S3Client
s3SyncClient) {
        try {
            Validate.notEmpty(bucketName, "The bucket name must not be null or an
empty string.", "");
            Validate.notEmpty(key, "The object key must not be null or an empty
string.", "");
            s3SyncClient.headObject(HeadObjectRequest.builder()
                .bucket(bucketName)
                .key(key)
                .build());
            return true;
        } catch (NoSuchKeyException e) {
            return false;
        }
    }
}

```

- Per i dettagli sull'API, consulta la sezione API Reference. [HeadObject](#) AWS SDK for Java 2.x

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Convertire il testo in voce e viceversa utilizzando un AWS SDK

L'esempio di codice seguente mostra come:

- Usa Amazon Polly per sintetizzare un file di input di testo semplice (UTF-8) in un file audio.

- Carica il file audio in un bucket Amazon S3.
- Utilizzare Amazon Transcribe per convertire il file audio in testo.
- Visualizzare il testo.

Rust

SDK per Rust

Usa Amazon Polly per sintetizzare un file di input di testo semplice (UTF-8) in un file audio, caricare il file audio in un bucket Amazon S3, utilizzare Amazon Transcribe per convertire il file audio in testo e visualizzare il testo.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su. [GitHub](#)

Servizi utilizzati in questo esempio

- Amazon Polly
- Simple Storage Service (Amazon S3)
- Amazon Transcribe

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Crea un URL predefinito per Amazon S3 utilizzando un AWS SDK

Gli esempi di codice seguenti mostrano come creare un URL prefirmato per Amazon S3 e caricare un oggetto.

.NET

SDK per .NET

Note

C'è altro su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Genera un URL prefirmato in grado di eseguire un'operazione Amazon S3 per un periodo di tempo limitato.

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

public class GenPresignedUrl
{
    public static void Main()
    {
        const string bucketName = "amzn-s3-demo-bucket";
        const string objectKey = "sample.txt";

        // Specify how long the presigned URL lasts, in hours
        const double timeoutDuration = 12;

        // Specify the AWS Region of your Amazon S3 bucket. If it is
        // different from the Region defined for the default user,
        // pass the Region to the constructor for the client. For
        // example: new AmazonS3Client(RegionEndpoint.USEast1);

        // If using the Region us-east-1, and server-side encryption with AWS
        KMS, you must specify Signature Version 4.
        // Region us-east-1 defaults to Signature Version 2 unless explicitly
        set to Version 4 as shown below.
        // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingAWSSDK.html#specify-signature-version
        // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/TAWSConfigsS3.html
        AWSConfigsS3.UseSignatureVersion4 = true;
        IAmazonS3 s3Client = new AmazonS3Client(RegionEndpoint.USEast1);

        string urlString = GeneratePresignedURL(s3Client, bucketName,
        objectKey, timeoutDuration);
        Console.WriteLine($"The generated URL is: {urlString}.");
    }

    /// <summary>
    /// Generate a presigned URL that can be used to access the file named
    /// in the objectKey parameter for the amount of time specified in the
    /// duration parameter.

```

```
/// </summary>
/// <param name="client">An initialized S3 client object used to call
/// the GetPresignedUrl method.</param>
/// <param name="bucketName">The name of the S3 bucket containing the
/// object for which to create the presigned URL.</param>
/// <param name="objectKey">The name of the object to access with the
/// presigned URL.</param>
/// <param name="duration">The length of time for which the presigned
/// URL will be valid.</param>
/// <returns>A string representing the generated presigned URL.</returns>
public static string GeneratePresignedURL(IAmazonS3 client, string
bucketName, string objectKey, double duration)
{
    string urlString = string.Empty;
    try
    {
        var request = new GetPreSignedUrlRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
            Expires = DateTime.UtcNow.AddHours(duration),
        };
        urlString = client.GetPreSignedURL(request);
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error: '{ex.Message}'");
    }

    return urlString;
}
}
```

Genera un URL prefirmato ed esegui un caricamento utilizzando quell'URL.

```
using System;
using System.IO;
using System.Net.Http;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
```

```
using Amazon.S3.Model;

/// <summary>
/// This example shows how to upload an object to an Amazon Simple Storage
/// Service (Amazon S3) bucket using a presigned URL. The code first
/// creates a presigned URL and then uses it to upload an object to an
/// Amazon S3 bucket using that URL.
/// </summary>
public class UploadUsingPresignedURL
{
    private static HttpClient httpClient = new HttpClient();

    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "samplefile.txt";
        string filePath = $"source\\{keyName}";

        // Specify how long the signed URL will be valid in hours.
        double timeoutDuration = 12;

        // Specify the AWS Region of your Amazon S3 bucket. If it is
        // different from the Region defined for the default user,
        // pass the Region to the constructor for the client. For
        // example: new AmazonS3Client(RegionEndpoint.USEast1);

        // If using the Region us-east-1, and server-side encryption with AWS
        KMS, you must specify Signature Version 4.
        // Region us-east-1 defaults to Signature Version 2 unless explicitly
        set to Version 4 as shown below.
        // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingAWSSDK.html#specify-signature-version
        // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/TAWSConfigsS3.html
        AWSConfigsS3.UseSignatureVersion4 = true;
        IAmazonS3 client = new AmazonS3Client(RegionEndpoint.USEast1);

        var url = GeneratePreSignedURL(client, bucketName, keyName,
            timeoutDuration);
        var success = await UploadObject(filePath, url);

        if (success)
        {
            Console.WriteLine("Upload succeeded.");
        }
    }
}
```

```

        }
        else
        {
            Console.WriteLine("Upload failed.");
        }
    }

    /// <summary>
    /// Uploads an object to an Amazon S3 bucket using the presigned URL
    passed in
    /// the url parameter.
    /// </summary>
    /// <param name="filePath">The path (including file name) to the local
    /// file you want to upload.</param>
    /// <param name="url">The presigned URL that will be used to upload the
    /// file to the Amazon S3 bucket.</param>
    /// <returns>A Boolean value indicating the success or failure of the
    /// operation, based on the HttpResponseMessage.</returns>
    public static async Task<bool> UploadObject(string filePath, string url)
    {
        using var streamContent = new StreamContent(
            new FileStream(filePath, FileMode.Open, FileAccess.Read));

        var response = await httpClient.PutAsync(url, streamContent);
        return response.IsSuccessStatusCode;
    }

    /// <summary>
    /// Generates a presigned URL which will be used to upload an object to
    /// an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
    call
    /// GetPreSignedURL.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket to which
    the
    /// presigned URL will point.</param>
    /// <param name="objectKey">The name of the file that will be uploaded.</
    param>
    /// <param name="duration">How long (in hours) the presigned URL will
    /// be valid.</param>
    /// <returns>The generated URL.</returns>
    public static string GeneratePreSignedURL(
        IAmazonS3 client,

```

```

        string bucketName,
        string objectKey,
        double duration)
    {
        var request = new GetPreSignedUrlRequest
        {
            BucketName = bucketName,
            Key = objectKey,
            Verb = HttpVerb.PUT,
            Expires = DateTime.UtcNow.AddHours(duration),
        };

        string url = client.GetPreSignedURL(request);
        return url;
    }
}

```

SDK per .NET (v4)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea e utilizza URL POST prefirmati per i caricamenti diretti dal browser.

```

/// <summary>
/// Scenario demonstrating the complete workflow for presigned POST URLs:
/// 1. Create an S3 bucket
/// 2. Create a presigned POST URL
/// 3. Upload a file using the presigned POST URL
/// 4. Clean up resources
/// </summary>
public class CreatePresignedPostBasics
{
    public static ILogger<CreatePresignedPostBasics> _logger = null!;
    public static S3Wrapper _s3Wrapper = null!;
    public static UiMethods _uiMethods = null!;
    public static IHttpClientFactory _httpClientFactory = null!;
    public static bool _isInteractive = true;
}

```



```
public static string? _bucketName;
public static string? _objectKey;

/// <summary>
/// Set up the services and logging.
/// </summary>
/// <param name="host">The IHost instance.</param>
public static void SetupServices(IHost host)
{
    var loggerFactory = LoggerFactory.Create(builder =>
    {
        builder.AddConsole();
    });
    _logger = new Logger<CreatePresignedPostBasics>(loggerFactory);

    _s3Wrapper = host.Services.GetRequiredService<S3Wrapper>();
    _httpClientFactory =
host.Services.GetRequiredService<IHttpClientFactory>();
    _uiMethods = new UiMethods();
}

/// <summary>
/// Perform the actions defined for the Amazon S3 Presigned POST scenario.
/// </summary>
/// <param name="args">Command line arguments.</param>
/// <returns>A Task object.</returns>
public static async Task Main(string[] args)
{
    _isInteractive = !args.Contains("--non-interactive");

    // Set up dependency injection for Amazon S3
    using var host =
Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
        .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonS3>()
                .AddTransient<S3Wrapper>()
                .AddHttpClient()
        )
        .Build();

    SetupServices(host);

    try
    {
```

```
// Display overview
_uiMethods.DisplayOverview();
_uiMethods.PressEnter(_isInteractive);

// Step 1: Create bucket
await CreateBucketAsync();
_uiMethods.PressEnter(_isInteractive);

// Step 2: Create presigned URL
_uiMethods.DisplayTitle("Step 2: Create presigned POST URL");
var response = await CreatePresignedPostAsync();
_uiMethods.PressEnter(_isInteractive);

// Step 3: Display URL and fields
_uiMethods.DisplayTitle("Step 3: Presigned POST URL details");
DisplayPresignedPostFields(response);
_uiMethods.PressEnter(_isInteractive);

// Step 4: Upload file
_uiMethods.DisplayTitle("Step 4: Upload test file using presigned
POST URL");
await UploadFileAsync(response);
_uiMethods.PressEnter(_isInteractive);

// Step 5: Verify file exists
await VerifyFileExistsAsync();
_uiMethods.PressEnter(_isInteractive);

// Step 6: Cleanup
_uiMethods.DisplayTitle("Step 6: Clean up resources");
await CleanupAsync();

_uiMethods.DisplayTitle("S3 Presigned POST Scenario completed
successfully!");
_uiMethods.PressEnter(_isInteractive);
}
catch (Exception ex)
{
    _logger.LogError(ex, "Error in scenario");
    Console.WriteLine($"Error: {ex.Message}");

    // Attempt cleanup if there was an error
    if (!string.IsNullOrEmpty(_bucketName))
    {
```

```

        _uiMethods.DisplayTitle("Cleaning up resources after error");
        await _s3Wrapper.DeleteBucketAsync(_bucketName);
        Console.WriteLine($"Cleaned up bucket: {_bucketName}");
    }
}

/// <summary>
/// Create an S3 bucket for the scenario.
/// </summary>
private static async Task CreateBucketAsync()
{
    _uiMethods.DisplayTitle("Step 1: Create an S3 bucket");

    // Generate a default bucket name for the scenario
    var defaultBucketName = $"presigned-post-demo-
{DateTime.Now:yyyyMMddHHmmss}".ToLower();

    // Prompt user for bucket name or use default in non-interactive mode
    _bucketName = _uiMethods.GetUserInput(
        $"Enter S3 bucket name (or press Enter for '{defaultBucketName}'):",
        defaultBucketName,
        _isInteractive);

    // Basic validation to ensure bucket name is not empty
    if (string.IsNullOrEmpty(_bucketName))
    {
        _bucketName = defaultBucketName;
    }

    Console.WriteLine($"Creating bucket: {_bucketName}");

    await _s3Wrapper.CreateBucketAsync(_bucketName);

    Console.WriteLine($"Successfully created bucket: {_bucketName}");
}

/// <summary>
/// Create a presigned POST URL.
/// </summary>
private static async Task<CreatePresignedPostResponse>
CreatePresignedPostAsync()
{

```

```
        _objectKey = "example-upload.txt";
        var expiration = DateTime.UtcNow.AddMinutes(10); // Short expiration for
the demo

        Console.WriteLine($"Creating presigned POST URL for {_bucketName}/
{_objectKey}");
        Console.WriteLine($"Expiration: {expiration} UTC");

        var s3Client = _s3Wrapper.GetS3Client();

        var response = await _s3Wrapper.CreatePresignedPostAsync(
            s3Client, _bucketName!, _objectKey, expiration);

        Console.WriteLine("Successfully created presigned POST URL");
        return response;
    }

    /// <summary>
    /// Upload a file using the presigned POST URL.
    /// </summary>
    private static async Task UploadFileAsync(CreatePresignedPostResponse
response)
    {

        // Create a temporary test file to upload
        string testFilePath = Path.GetTempFileName();
        string testContent = "This is a test file for the S3 presigned POST
scenario.";

        await File.WriteAllTextAsync(testFilePath, testContent);
        Console.WriteLine($"Created test file at: {testFilePath}");

        // Upload the file using the presigned POST URL
        Console.WriteLine("\nUploading file using the presigned POST URL...");
        var uploadResult = await UploadFileWithPresignedPostAsync(response,
testFilePath);

        // Display the upload result
        if (uploadResult.Success)
        {
            Console.WriteLine($"Upload successful! Status code:
{uploadResult.StatusCode}");
        }
        else
```

```
        {
            Console.WriteLine($"Upload failed with status code:
{uploadResult.StatusCode}");
            Console.WriteLine($"Error: {uploadResult.Response}");
            throw new Exception("File upload failed");
        }

        // Clean up the temporary file
        File.Delete(testFilePath);
        Console.WriteLine("Temporary file deleted");
    }

    /// <summary>
    /// Helper method to upload a file using a presigned POST URL.
    /// </summary>
    private static async Task<(bool Success, HttpStatusCode StatusCode, string
Response)> UploadFileWithPresignedPostAsync(
        CreatePresignedPostResponse response,
        string filePath)
    {
        try
        {
            _logger.LogInformation("Uploading file {filePath} using presigned
POST URL", filePath);

            using var httpClient = _httpClientFactory.CreateClient();
            using var formContent = new MultipartFormDataContent();

            // Add all the fields from the presigned POST response
            foreach (var field in response.Fields)
            {
                formContent.Add(new StringContent(field.Value), field.Key);
            }

            // Add the file content
            var fileStream = File.OpenRead(filePath);
            var fileName = Path.GetFileName(filePath);
            var fileContent = new StreamContent(fileStream);
            fileContent.Headers.ContentType = new MediaTypeHeaderValue("text/
plain");
            formContent.Add(fileContent, "file", fileName);

            // Send the POST request
```

```
        var httpResponse = await httpClient.PostAsync(response.Url,
formContent);
        var responseContent = await httpResponse.Content.ReadAsStringAsync();

        // Log and return the result
        _logger.LogInformation("Upload completed with status code
{statusCode}", httpResponse.StatusCode);

        return (httpResponse.IsSuccessStatusCode, httpResponse.StatusCode,
responseContent);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Error uploading file");
        return (false, HttpStatusCode.InternalServerError, ex.Message);
    }
}

/// <summary>
/// Verify that the uploaded file exists in the S3 bucket.
/// </summary>
private static async Task VerifyFileExistsAsync()
{
    _uiMethods.DisplayTitle("Step 5: Verify uploaded file exists");

    Console.WriteLine($"Checking if file exists at {_bucketName}/
{_objectKey}...");

    try
    {
        var metadata = await _s3Wrapper.GetObjectMetadataAsync(_bucketName!,
_objectKey!);

        Console.WriteLine($"File verification successful! File exists in the
bucket.");
        Console.WriteLine($"File size: {metadata.ContentLength} bytes");
        Console.WriteLine($"File type: {metadata.Headers.ContentType}");
        Console.WriteLine($"Last modified: {metadata.LastModified}");
    }
    catch (AmazonS3Exception ex) when (ex.StatusCode ==
System.Net.HttpStatusCode.NotFound)
    {
        Console.WriteLine($"Error: File was not found in the bucket.");
        throw;
    }
}
```

```
    }
}

private static void DisplayPresignedPostFields(CreatePresignedPostResponse
response)
{
    Console.WriteLine($"Presigned POST URL: {response.Url}");
    Console.WriteLine("Form fields to include:");

    foreach (var field in response.Fields)
    {
        Console.WriteLine($"  {field.Key}: {field.Value}");
    }
}

/// <summary>
/// Clean up resources created by the scenario.
/// </summary>
private static async Task CleanupAsync()
{
    if (!string.IsNullOrEmpty(_bucketName))
    {
        Console.WriteLine($"Deleting bucket {_bucketName} and its
contents...");
        bool result = await _s3Wrapper.DeleteBucketAsync(_bucketName);

        if (result)
        {
            Console.WriteLine("Bucket deleted successfully");
        }
        else
        {
            Console.WriteLine("Failed to delete bucket - it may have been
already deleted");
        }
    }
}
}
```

C++

SDK per C++

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Genera un URL prefirmato per scaricare un oggetto.

```
//! Routine which demonstrates creating a pre-signed URL to download an object
    from an
    //! Amazon Simple Storage Service (Amazon S3) bucket.
    /*!
        \param bucketName: Name of the bucket.
        \param key: Name of an object key.
        \param expirationSeconds: Expiration in seconds for pre-signed URL.
        \param clientConfig: Aws client configuration.
        \return Aws::String: A pre-signed URL.
    */
    Aws::String AwsDoc::S3::generatePreSignedGetObjectUrl(const Aws::String
        &bucketName,
                                                    const Aws::String &key,
                                                    uint64_t expirationSeconds,
                                                    const
        Aws::S3::S3ClientConfiguration &clientConfig) {
        Aws::S3::S3Client client(clientConfig);
        return client.GeneratePresignedUrl(bucketName, key,
        Aws::Http::HttpMethod::HTTP_GET,
                                                    expirationSeconds);
    }
```

Esegue scaricamenti tramite libcurl.

```
static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
    *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
```



```
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test getObject with a pre-signed URL.
/*!
    \param presignedURL: A pre-signed URL to get an object from a bucket.
    \param resultString: A string to hold the result.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::getObjectWithPresignedObjectUrl(const Aws::String &presignedURL,
                                                  Aws::String &resultString) {

    CURL *curl = curl_easy_init();
    CURLcode result;

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }
}
```

```

    }

    resultString = outWriteString.str();

    if (resultString.find("<?xml") == 0) {
        std::cerr << "Failed to get object, response:\n" << resultString <<
std::endl;
        return false;
    }

    return true;
}

```

Genera un URL prefirmato per caricare un oggetto.

```

//! Routine which demonstrates creating a pre-signed URL to upload an object to
an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
    \param bucketName: Name of the bucket.
    \param key: Name of an object key.
    \param clientConfig: Aws client configuration.
    \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedPutObjectUrl(const Aws::String
&bucketName,
                                                    const Aws::String &key,
                                                    uint64_t expirationSeconds,
                                                    const
Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
    Aws::Http::HttpMethod::HTTP_PUT,
                                expirationSeconds);
}

```

Esegue caricamenti tramite libcurl.

```

static size_t myCurlReadBack(char *buffer, size_t size, size_t nitems, void
*userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

```

```

        str->read(buffer, size * nitems);

        return str->gcount();
    }

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
*userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test putObject with a pre-signed URL.
/*!
    \param presignedURL: A pre-signed URL to put an object in a bucket.
    \param data: Body of the putObject request.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::PutStringWithPresignedObjectURL(const Aws::String &presignedURL,
                                                  const Aws::String &data) {

    CURL *curl = curl_easy_init();
    CURLcode result;

    Aws::StringStream readStringStream;
    readStringStream << data;
    result = curl_easy_setopt(curl, CURLOPT_READFUNCTION, myCurlReadBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_READDATA, &readStringStream);
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READDATA" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_INFILESIZE_LARGE,
                              (curl_off_t) data.size());

```

```
if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_INFILESIZE_LARGE" << std::endl;
    return false;
}

result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
    return false;
}

std::stringstream outWriteString;

result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
    return false;
}

result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_URL" << std::endl;
    return false;
}

result = curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_PUT" << std::endl;
    return false;
}

result = curl_easy_perform(curl);

if (result != CURLE_OK) {
    std::cerr << "Failed to perform CURL request" << std::endl;
    return false;
}

std::string outString = outWriteString.str();
```

```
    if (outString.empty()) {
        std::cout << "Successfully put object." << std::endl;
        return true;
    } else {
        std::cout << "A server error was encountered, output:\n" << outString
            << std::endl;
        return false;
    }
}
```

Go

SDK per Go V2

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea funzioni che eseguono il wrap delle operazioni S3 di prefirma.

```
import (
    "context"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    v4 "github.com/aws/aws-sdk-go-v2/aws/signer/v4"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)

// Presigner encapsulates the Amazon Simple Storage Service (Amazon S3) presign
// actions
// used in the examples.
// It contains PresignClient, a client that is used to presign requests to Amazon
// S3.
// Presigned requests contain temporary credentials and can be made from any HTTP
// client.
type Presigner struct {
```

```
PresignClient *s3.PresignClient
}

// GetObject makes a presigned request that can be used to get an object from a
// bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) GetObject(
    ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
(*v4.PresignedHTTPRequest, error) {
    request, err := presigner.PresignClient.PresignGetObject(ctx,
        &s3.GetObjectInput{
            Bucket: aws.String(bucketName),
            Key:    aws.String(objectKey),
        }, func(opts *s3.PresignOptions) {
            opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
        })
    if err != nil {
        log.Printf("Couldn't get a presigned request to get %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return request, err
}

// PutObject makes a presigned request that can be used to put an object in a
// bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) PutObject(
    ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
(*v4.PresignedHTTPRequest, error) {
    request, err := presigner.PresignClient.PresignPutObject(ctx,
        &s3.PutObjectInput{
            Bucket: aws.String(bucketName),
            Key:    aws.String(objectKey),
        }, func(opts *s3.PresignOptions) {
            opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
        })
    if err != nil {
        log.Printf("Couldn't get a presigned request to put %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
}
```

```
    return request, err
}

// DeleteObject makes a presigned request that can be used to delete an object
// from a bucket.
func (presigner Presigner) DeleteObject(ctx context.Context, bucketName string,
    objectKey string) (*v4.PresignedHTTPRequest, error) {
    request, err := presigner.PresignClient.PresignDeleteObject(ctx,
        &s3.DeleteObjectInput{
            Bucket: aws.String(bucketName),
            Key:    aws.String(objectKey),
        })
    if err != nil {
        log.Printf("Couldn't get a presigned request to delete object %v. Here's why:
        %v\n", objectKey, err)
    }
    return request, err
}

func (presigner Presigner) PresignPostObject(ctx context.Context, bucketName
    string, objectKey string, lifetimeSecs int64) (*s3.PresignedPostRequest, error)
{
    request, err := presigner.PresignClient.PresignPostObject(ctx,
        &s3.PutObjectInput{
            Bucket: aws.String(bucketName),
            Key:    aws.String(objectKey),
        }, func(options *s3.PresignPostOptions) {
            options.Expires = time.Duration(lifetimeSecs) * time.Second
        })
    if err != nil {
        log.Printf("Couldn't get a presigned post request to put %v:%v. Here's why: %v
        \n", bucketName, objectKey, err)
    }
    return request, nil
}
```

Esegui un esempio interattivo che genera e utilizza URL prefirmati per caricare, scaricare ed eliminare un oggetto S3.

```
import (  
    "bytes"  
    "context"  
    "io"  
    "log"  
    "mime/multipart"  
    "net/http"  
    "os"  
    "strings"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"  
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"  
)  
  
// RunPresigningScenario is an interactive example that shows you how to get  
// presigned  
// HTTP requests that you can use to move data into and out of Amazon Simple  
// Storage  
// Service (Amazon S3). The presigned requests contain temporary credentials and  
// can  
// be used by an HTTP client.  
//  
// 1. Get a presigned request to put an object in a bucket.  
// 2. Use the net/http package to use the presigned request to upload a local  
//    file to the bucket.  
// 3. Get a presigned request to get an object from a bucket.  
// 4. Use the net/http package to use the presigned request to download the  
//    object to a local file.  
// 5. Get a presigned request to delete an object from a bucket.  
// 6. Use the net/http package to use the presigned request to delete the object.  
//  
// This example creates an Amazon S3 presign client from the specified sdkConfig  
// so that  
// you can replace it with a mocked or stubbed config for unit testing.  
//
```



```
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../../demotools folder of this repo.
//
// It uses an IHttpRequester interface to abstract HTTP requests so they can be
// mocked
// during testing.
func RunPresigningScenario(ctx context.Context, sdkConfig aws.Config, questioner
demotools.IQuestioner, httpRequester IHttpRequester) {
defer func() {
    if r := recover(); r != nil {
        log.Println("Something went wrong with the demo.")
        _, isMock := questioner.(*demotools.MockQuestioner)
        if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
            log.Println(r)
        }
    }
}()

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon S3 presigning demo.")
log.Println(strings.Repeat("-", 88))

s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}
presignClient := s3.NewPresignClient(s3Client)
presigner := actions.Presigner{PresignClient: presignClient}

bucketName := questioner.Ask("We'll need a bucket. Enter a name for a bucket "+
    "you own or one you want to create:", demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
    panic(err)
}
if !bucketExists {
    err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
    if err != nil {
        panic(err)
    } else {
        log.Println("Bucket created.")
    }
}
log.Println(strings.Repeat("-", 88))
```

```
log.Printf("Let's presign a request to upload a file to your bucket.")
uploadFilename := questioner.Ask("Enter the path to a file you want to upload:",
    demotools.NotEmpty{})
uploadKey := questioner.Ask("What would you like to name the uploaded object?",
    demotools.NotEmpty{})
uploadFile, err := os.Open(uploadFilename)
if err != nil {
    panic(err)
}
defer uploadFile.Close()
presignedPutRequest, err := presigner.PutObject(ctx, bucketName, uploadKey, 60)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
    presignedPutRequest.Method,
    presignedPutRequest.URL)
log.Println("Using net/http to send the request...")
info, err := uploadFile.Stat()
if err != nil {
    panic(err)
}
putResponse, err := httpRequester.Put(presignedPutRequest.URL, info.Size(),
    uploadFile)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
    presignedPutRequest.Method,
    uploadKey, putResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to download the object.")
questioner.Ask("Press Enter when you're ready.")
presignedGetRequest, err := presigner.GetObject(ctx, bucketName, uploadKey, 60)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
    presignedGetRequest.Method,
    presignedGetRequest.URL)
log.Println("Using net/http to send the request...")
getResponse, err := httpRequester.Get(presignedGetRequest.URL)
```

```
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedGetRequest.Method,
uploadKey, getResponse.StatusCode)
defer getResponse.Body.Close()
downloadBody, err := io.ReadAll(getResponse.Body)
if err != nil {
    panic(err)
}
log.Printf("Downloaded %v bytes. Here are the first 100 of them:\n",
len(downloadBody))
log.Println(strings.Repeat("-", 88))
log.Println(string(downloadBody[:100]))
log.Println(strings.Repeat("-", 88))

log.Println("Now we'll create a new request to put the same object using a
presigned post request")
questioner.Ask("Press Enter when you're ready.")
presignPostRequest, err := presigner.PresignPostObject(ctx, bucketName,
uploadKey, 60)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned post request to url %v with values %v\n",
presignPostRequest.URL, presignPostRequest.Values)
log.Println("Using net/http multipart to send the request...")
uploadFile, err = os.Open(uploadFilename)
if err != nil {
    panic(err)
}
defer uploadFile.Close()
multiPartResponse, err := sendMultipartRequest(presignPostRequest.URL,
presignPostRequest.Values, uploadFile, uploadKey, httpRequester)
if err != nil {
    panic(err)
}
log.Printf("Presign post object %v with presigned URL returned %v.", uploadKey,
multiPartResponse.StatusCode)

log.Println("Let's presign a request to delete the object.")
questioner.Ask("Press Enter when you're ready.")
presignedDelRequest, err := presigner.DeleteObject(ctx, bucketName, uploadKey)
```

```
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedDelRequest.Method,
presignedDelRequest.URL)
log.Println("Using net/http to send the request...")
delResponse, err := httpRequester.Delete(presignedDelRequest.URL)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.\n",
presignedDelRequest.Method,
uploadKey, delResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

Definisci un wrapper di richieste HTTP utilizzato dall'esempio per effettuare richieste HTTP.

```
// IHttpRequester abstracts HTTP requests into an interface so it can be mocked
// during
// unit testing.
type IHttpRequester interface {
    Get(url string) (resp *http.Response, err error)
    Post(url, contentType string, body io.Reader) (resp *http.Response, err error)
    Put(url string, contentLength int64, body io.Reader) (resp *http.Response, err
    error)
    Delete(url string) (resp *http.Response, err error)
}

// HttpRequester uses the net/http package to make HTTP requests during the
// scenario.
type HttpRequester struct{}

func (httpReq HttpRequester) Get(url string) (resp *http.Response, err error) {
    return http.Get(url)
}
```

```
func (httpReq HttpRequester) Post(url, contentType string, body io.Reader) (resp
    *http.Response, err error) {
    postRequest, err := http.NewRequest("POST", url, body)
    if err != nil {
        return nil, err
    }
    postRequest.Header.Set("Content-Type", contentType)
    return http.DefaultClient.Do(postRequest)
}

func (httpReq HttpRequester) Put(url string, contentLength int64, body io.Reader)
    (resp *http.Response, err error) {
    putRequest, err := http.NewRequest("PUT", url, body)
    if err != nil {
        return nil, err
    }
    putRequest.ContentLength = contentLength
    return http.DefaultClient.Do(putRequest)
}

func (httpReq HttpRequester) Delete(url string) (resp *http.Response, err error)
    {
    delRequest, err := http.NewRequest("DELETE", url, nil)
    if err != nil {
        return nil, err
    }
    return http.DefaultClient.Do(delRequest)
}
```

Java

SDK per Java 2.x

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Di seguito vengono mostrati tre esempi di come creare URL prefirmati e utilizzare gli URL con le librerie client HTTP:

- Una richiesta HTTP GET che utilizza l'URL con tre librerie client HTTP
- Una richiesta HTTP PUT con metadati nelle intestazioni, che utilizza l'URL con tre librerie client HTTP
- Una richiesta HTTP PUT con parametri di query che utilizza l'URL con una libreria client HTTP

Genera un URL prefirmato per un oggetto, quindi lo scarica (richiesta GET).

Esegue l'importazione.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
```

```
import java.util.UUID;
```

Genera l'URL.

```
/* Create a pre-signed URL to download an object in a subsequent GET request.
*/
public String createPresignedGetUrl(String bucketName, String keyName) {
    try (S3Presigner presigner = S3Presigner.create()) {

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        GetObjectPresignRequest presignRequest =
            GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(10)) // The URL will
                expire in 10 minutes.
                .getObjectRequest(objectRequest)
                .build();

        PresignedGetObjectRequest presignedRequest =
            presigner.presignGetObject(presignRequest);
        logger.info("Presigned URL: [{}]",
            presignedRequest.url().toString());
        logger.info("HTTP method: [{}]",
            presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}
```

Scarica l'oggetto utilizzando uno dei seguenti tre approcci.

Utilizza la classe `HttpURLConnection` di JDK (dalla v1.1) per eseguire lo scaricamento.

```
/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useHttpURLConnectionToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
        ByteArrayOutputStream(); // Capture the response body to a byte array.
```

```
try {
    URL presignedUrl = new URL(presignedUrlString);
    HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
    connection.setRequestMethod("GET");
    // Download the result of executing the request.
    try (InputStream content = connection.getInputStream()) {
        IoUtils.copy(content, byteArrayOutputStream);
    }
    logger.info("HTTP response code is " + connection.getResponseCode());

} catch (S3Exception | IOException e) {
    logger.error(e.getMessage(), e);
}
return byteArrayOutputStream.toByteArray();
}
```

Utilizza la classe `HttpClient` di JDK (dalla v11) per eseguire lo scaricamento.

```
/* Use the JDK HttpClient (since v11) class to do the download. */
public byte[] useHttpClientToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpResponse<InputStream> response = httpClient.send(requestBuilder
            .uri(presignedUrl.toURI())
            .GET()
            .build(),
            HttpResponse.BodyHandlers.ofInputStream());

        IoUtils.copy(response.body(), byteArrayOutputStream);

        logger.info("HTTP response code is " + response.statusCode());

    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}
```



```
}
```

Utilizzate la classe AWS SDK for SdkHttpClient Java per eseguire il download.

```
/* Use the AWS SDK for Java SdkHttpClient class to do the download. */
public byte[] useSdkHttpClientToGet(String presignedUrlString) {

    ByteArrayOutputStream byteArrayOutputStream = new
    ByteArrayOutputStream(); // Capture the response body to a byte array.
    try {
        URL presignedUrl = new URL(presignedUrlString);
        SdkHttpRequest request = SdkHttpRequest.builder()
            .method(SdkHttpMethod.GET)
            .uri(presignedUrl.toURI())
            .build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
            sdkHttpClient.prepareRequest(executeRequest).call();
            response.responseBody().ifPresentOrElse(
                abortableInputStream -> {
                    try {
                        IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                },
                () -> logger.error("No response body."));

            logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}
```

Genera un URL prefirmato con metadati nelle intestazioni per un caricamento, quindi carica un file (richiesta PUT).

Esegue l'importazione.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

Genera l'URL.

```
/* Create a presigned URL to use in a subsequent PUT request */
public String createPresignedUrl(String bucketName, String keyName,
    Map<String, String> metadata) {
    try (S3Presigner presigner = S3Presigner.create()) {

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .metadata(metadata)
            .build();

        PutObjectPresignRequest presignRequest =
            PutObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(10)) // The URL
                expires in 10 minutes.
                .putObjectRequest(objectRequest)
                .build();

        PresignedPutObjectRequest presignedRequest =
            presigner.presignPutObject(presignRequest);
        String myURL = presignedRequest.url().toString();
        logger.info("Presigned URL to upload a file to: [{}]", myURL);
        logger.info("HTTP method: [{}]",
            presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}
```

Carica un oggetto file utilizzando uno dei tre approcci seguenti.

Utilizza la classe JDK `URLConnection` (dalla v1.1) per eseguire il caricamento.

```
/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
public void useHttpURLConnectionToPut(String presignedUrlString, File
    fileToPut, Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());
    try {
        URL presignedUrl = new URL(presignedUrlString);
```

```

        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setDoOutput(true);
        metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-
meta-" + k, v));
        connection.setRequestMethod("PUT");
        OutputStream out = connection.getOutputStream();

        try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
            FileChannel inChannel = file.getChannel()) {
            ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is
8k

            while (inChannel.read(buffer) > 0) {
                buffer.flip();
                for (int i = 0; i < buffer.limit(); i++) {
                    out.write(buffer.get());
                }
                buffer.clear();
            }
        } catch (IOException e) {
            logger.error(e.getMessage(), e);
        }

        out.close();
        connection.getResponseCode();
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

```

Utilizza la classe JDK `HttpClient` (dalla v11) per eseguire il caricamento.

```

/* Use the JDK HttpClient (since v11) class to do the upload. */
public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));
}

```

```

HttpClient httpClient = HttpClient.newHttpClient();
try {
    final HttpResponse<Void> response = httpClient.send(requestBuilder
        .uri(new URL(presignedUrlString).toURI())

    .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI()))
        .build(),
        HttpResponse.BodyHandlers.discarding());

    logger.info("HTTP response code is " + response.statusCode());

} catch (URISyntaxException | InterruptedException | IOException e) {
    logger.error(e.getMessage(), e);
}
}

```

Utilizzate la `SdkHttpClient` classe AWS for Java V2 per eseguire il caricamento.

```

/* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
    Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);

        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());
        // Add headers
        metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" +
k, v));
        // Finish building the request.
        SdkHttpRequest request = requestBuilder.build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
            .build();

```

```

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            logger.info("Response code: {}",
response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

```

Genera un URL prefirmato con metadati di query per un caricamento, quindi carica un file (richiesta PUT).

Esegue l'importazione.

```

import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.awscore.AwsRequestOverrideConfiguration;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;

```

Genera l'URL.

```
/**
 * Creates a presigned URL to use in a subsequent HTTP PUT request. The code
 adds query parameters
 * to the request instead of using headers. By using query parameters, you
 do not need to add the
 * the parameters as headers when the PUT request is eventually sent.
 *
 * @param bucketName Bucket name where the object will be uploaded.
 * @param keyName Key name of the object that will be uploaded.
 * @param queryParams Query string parameters to be added to the presigned
URL.
 * @return
 */
public String createPresignedUrl(String bucketName, String keyName,
Map<String, String> queryParams) {
    try (S3Presigner presigner = S3Presigner.create()) {
        // Create an override configuration to store the query parameters.
        AwsRequestOverrideConfiguration.Builder overrideConfigurationBuilder
= AwsRequestOverrideConfiguration.builder();

queryParams.forEach(overrideConfigurationBuilder::putRawQueryParameter);

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)

.overrideConfiguration(overrideConfigurationBuilder.build()) // Add the override
configuration.

            .build();

        PutObjectPresignRequest presignRequest =
PutObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL
expires in 10 minutes.

            .putObjectRequest(objectRequest)
            .build();
```

```

        PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
        String myURL = presignedRequest.url().toString();
        logger.info("Presigned URL to upload a file to: [{}]", myURL);
        logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

Utilizzate la `SdkHttpClient` classe AWS for Java V2 per eseguire il caricamento.

```

/**
 * Use the AWS SDK for Java V2 SdkHttpClient class to execute the PUT
request. Since the
 * URL contains the query parameters, no headers are needed for metadata, SSE
settings, or ACL settings.
 *
 * @param presignedUrlString The URL for the PUT request.
 * @param fileToPut File to upload
 */
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut)
{
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);

        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());

        SdkHttpRequest request = requestBuilder.build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {

```



```
        HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
        logger.info("Response code: {}",
response.httpResponse().statusCode());
    }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}
```

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea un URL prefirmato per caricare un oggetto in un bucket.

```
import https from "node:https";

import { XMLParser } from "fast-xml-parser";
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
    getSignedUrl,
    S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
    const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
    const presigner = new S3RequestPresigner({
        credentials: fromIni(),
        region,
```

```
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(
    new HttpRequest({ ...url, method: "PUT" }),
  );
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new PutObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

/**
 * Make a PUT request to the provided URL.
 *
 * @param {string} url
 * @param {string} data
 */
const put = (url, data) => {
  return new Promise((resolve, reject) => {
    const req = https.request(
      url,
      { method: "PUT", headers: { "Content-Length": new Blob([data]).size } },
      (res) => {
        let responseBody = "";
        res.on("data", (chunk) => {
          responseBody += chunk;
        });
        res.on("end", () => {
          const parser = new XMLParser();
          if (res.statusCode >= 200 && res.statusCode <= 299) {
            resolve(parser.parse(responseBody, true));
          } else {
            reject(parser.parse(responseBody, true));
          }
        });
      },
    );
    req.on("error", (err) => {
      reject(err);
    });
  });
};
```

```
    req.write(data);
    req.end();
  });
};

/**
 * Create two presigned urls for uploading an object to an S3 bucket.
 * The first presigned URL is created with credentials from the shared INI file
 * in the current environment. The second presigned URL is created using an
 * existing S3Client instance that has already been provided with credentials.
 * @param {{ bucketName: string, key: string, region: string }}
 */
export const main = async ({ bucketName, key, region }) => {
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      bucket: bucketName,
      key,
      region,
    });

    const clientUrl = await createPresignedUrlWithClient({
      bucket: bucketName,
      region,
      key,
    });

    // After you get the presigned URL, you can provide your own file
    // data. Refer to put() above.
    console.log("Calling PUT using presigned URL without client");
    await put(noClientUrl, "Hello World");

    console.log("Calling PUT using presigned URL with client");
    await put(clientUrl, "Hello World");

    console.log("\nDone. Check your S3 console.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "CredentialsProviderError") {
      console.error(
        `There was an error getting your credentials. Are your local credentials configured?\n${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
}
```

```

    }
};

```

Crea un URL prefirmato per scaricare un oggetto da un bucket.

```

import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
    getSignedUrl,
    S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
    const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
    const presigner = new S3RequestPresigner({
        credentials: fromIni(),
        region,
        sha256: Hash.bind(null, "sha256"),
    });

    const signedUrlObject = await presigner.presign(new HttpRequest(url));
    return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
    const client = new S3Client({ region });
    const command = new GetObjectCommand({ Bucket: bucket, Key: key });
    return getSignedUrl(client, command, { expiresIn: 3600 });
};

/**
 * Create two presigned urls for downloading an object from an S3 bucket.
 * The first presigned URL is created with credentials from the shared INI file
 * in the current environment. The second presigned URL is created using an
 * existing S3Client instance that has already been provided with credentials.
 * @param {{ bucketName: string, key: string, region: string }}
 */
export const main = async ({ bucketName, key, region }) => {

```

```
try {
  const noClientUrl = await createPresignedUrlWithoutClient({
    bucket: bucketName,
    region,
    key,
  });

  const clientUrl = await createPresignedUrlWithClient({
    bucket: bucketName,
    region,
    key,
  });

  console.log("Presigned URL without client");
  console.log(noClientUrl);
  console.log("\n");

  console.log("Presigned URL with client");
  console.log(clientUrl);
} catch (caught) {
  if (caught instanceof Error && caught.name === "CredentialsProviderError") {
    console.error(
      `There was an error getting your credentials. Are your local credentials configured?\n${caught.name}: ${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK per JavaScript](#).

Kotlin

SDK per Kotlin

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea una richiesta prefirmata `GetObject` e usa l'URL per scaricare un oggetto.

```
suspend fun getObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): String {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)

    // Use the URL from the presigned HttpRequest in a subsequent HTTP GET
    request to retrieve the object.
    val objectContents = URL(presignedRequest.url.toString()).readText()

    return objectContents
}
```

Crea una richiesta `GetObject` prefirmata con opzioni avanzate.

```
suspend fun getObjectPresignedMoreOptions(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): HttpRequest {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest =
        s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
```

```

        signingDate = Instant.now() + 12.hours // Presigned request can be
        used 12 hours from now.
        algorithm = AwsSigningAlgorithm.SIGV4_ASYMMETRIC
        signatureType = AwsSignatureType.HTTP_REQUEST_VIA_QUERY_PARAMS
        expiresAfter = 8.hours // Presigned request expires 8 hours later.
    }
    return presignedRequest
}

```

Crea una richiesta prefirmata `PutObject` e usala per caricare un oggetto.

```

suspend fun putObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
    content: String,
) {
    // Create a PutObjectRequest.
    val unsignedRequest =
        PutObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the request.
    val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)

    // Use the URL and any headers from the presigned HttpRequest in a subsequent
    HTTP PUT request to retrieve the object.
    // Create a PUT request using the OKHttpClient API.
    val putRequest =
        Request
            .Builder()
            .url(presignedRequest.url.toString())
            .apply {
                presignedRequest.headers.forEach { key, values ->
                    header(key, values.joinToString(", "))
                }
            }
            .put(content.toRequestBody())
            .build()

    val response = OkHttpClient().newCall(putRequest).execute()
}

```

```
    assert(response.isSuccessful)
}
```

- Per ulteriori informazioni, consulta la [Guida per gli sviluppatori di AWS SDK per Swift](#).

PHP

SDK per PHP

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
namespace S3;
use Aws\Exception\AwsException;
use AwsUtilities\PrintableLineBreak;
use AwsUtilities\TestableReadline;
use DateTime;

require 'vendor/autoload.php';

class PresignedURL
{
    use PrintableLineBreak;
    use TestableReadline;

    public function run()
    {
        $s3Service = new S3Service();

        $expiration = new DateTime("+20 minutes");
        $linebreak = $this->getLineBreak();

        echo $linebreak;
        echo ("Welcome to the Amazon S3 presigned URL demo.\n");
        echo $linebreak;

        $bucket = $this->testable_readline("First, please enter the name of the
S3 bucket to use: ");
```



```

        $key = $this->testable_readline("Next, provide the key of an object in
the given bucket: ");
        echo $linebreak;
        $command = $s3Service->getClient()->getCommand('GetObject', [
            'Bucket' => $bucket,
            'Key' => $key,
        ]);
        try {
            $preSignedUrl = $s3Service->preSignedUrl($command, $expiration);
            echo "Your preSignedUrl is \n$preSignedUrl\nand will be good for the
next 20 minutes.\n";
            echo $linebreak;
            echo "Thanks for trying the Amazon S3 presigned URL demo.\n";
        } catch (AwsException $exception) {
            echo $linebreak;
            echo "Something went wrong: $exception";
            die();
        }
    }
}

$runner = new PresignedURL();
$runner->run();

namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {

```

```
        $this->client = $client;
    } else {
        $this->client = new S3Client([
            'version' => 'latest',
            'region' => 'us-west-2',
        ]);
    }
    $this->verbose = $verbose;
}

public function setVerbose($verbose)
{
    $this->verbose = $verbose;
}

public function isVerbose(): bool
{
    return $this->verbose;
}

public function getClient(): S3Client
{
    return $this->client;
}

public function setClient(S3Client $client)
{
    $this->client = $client;
}

public function emptyAndDeleteBucket($bucketName, array $args = [])
{
    try {
        $objects = $this->listAllObjects($bucketName, $args);
        $this->deleteObjects($bucketName, $objects, $args);
        if ($this->verbose) {
            echo "Deleted all objects and folders from $bucketName.\n";
        }
        $this->deleteBucket($bucketName, $args);
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
        }
    }
}
```

```

        echo "\nPlease fix error with bucket deletion before continuing.
\n";
    }
    throw $exception;
}
}

public function createBucket(string $bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);
    try {
        $this->client->createBucket($parameters);
        if ($this->verbose) {
            echo "Created the bucket named: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket creation before continuing.";
        }
        throw $exception;
    }
}

public function putObject(string $bucketName, string $key, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
    try {
        $this->client->putObject($parameters);
        if ($this->verbose) {
            echo "Uploaded the object named: $key to the bucket named:
$bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create $key in $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object uploading before continuing.";
        }
    }
}

```

```

        }
        throw $exception;
    }
}

public function getObject(string $bucketName, string $key, array $args = []):
Result
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
    try {
        $object = $this->client->getObject($parameters);
        if ($this->verbose) {
            echo "Downloaded the object named: $key to the bucket named:
$bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to download $key from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object downloading before
continuing.";
        }
        throw $exception;
    }
    return $object;
}

public function copyObject($bucketName, $key, $copySource, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
"CopySource" => $copySource], $args);
    try {
        $this->client->copyObject($parameters);
        if ($this->verbose) {
            echo "Copied the object from: $copySource in $bucketName to:
$key.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {

```

```

        echo "Failed to copy $copySource in $bucketName with error:
{$exception->getMessage()}\n";
        echo "Please fix error with object copying before continuing.";
    }
    throw $exception;
}
}

public function listObjects(string $bucketName, $start = 0, $max = 1000,
array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
    try {
        $objects = $this->client->listObjectsV2($parameters);
        if ($this->verbose) {
            echo "Retrieved the list of objects from: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with list objects before continuing.";
        }
        throw $exception;
    }
    return $objects;
}

public function listAllObjects($bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);

    $contents = [];
    $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

    foreach ($paginator as $result) {
        if($result['KeyCount'] == 0){
            break;
        }
    }
}

```

```

        foreach ($result['Contents'] as $object) {
            $contents[] = $object;
        }
    }
    return $contents;
}

public function deleteObjects(string $bucketName, array $objects, array $args
= [])
{
    $listOfObjects = array_map(
        function ($object) {
            return ['Key' => $object];
        },
        array_column($objects, 'Key')
    );
    if(!$listOfObjects){
        return;
    }

    $parameters = array_merge(['Bucket' => $bucketName, 'Delete' =>
['Objects' => $listOfObjects]], $args);
    try {
        $this->client->deleteObjects($parameters);
        if ($this->verbose) {
            echo "Deleted the list of objects from: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete the list of objects from $bucketName with
error: {$exception->getMessage()}\n";
            echo "Please fix error with object deletion before continuing.";
        }
        throw $exception;
    }
}

public function deleteBucket(string $bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);

```

```

        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket deletion before continuing.";
            }
            throw $exception;
        }
    }

    public function deleteObject(string $bucketName, string $fileName, array
$args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
$args);
        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }

    public function listBuckets(array $args = [])
    {
        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {

```

```

        echo "Retrieved all " . count($buckets) . "\n";
    }
} catch (AwsException $exception) {
    if ($this->verbose) {
        echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
        echo "Please fix error with bucket lists before continuing.";
    }
    throw $exception;
}
return $buckets;
}

public function preSignedUrl(CommandInterface $command, DateTimeInterface|
int|string $expires, array $options = [])
{
    $request = $this->client->createPresignedRequest($command, $expires,
$options);
    try {
        $presignedUrl = (string)$request->getUri();
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
            echo "Please fix error with presigned urls before continuing.";
        }
        throw $exception;
    }
    return $presignedUrl;
}

public function createSession(string $bucketName)
{
    try{
        $result = $this->client->createSession([
            'Bucket' => $bucketName,
        ]);
        return $result;
    }catch(S3Exception $caught){
        if($caught->getAwsErrorType() == "NoSuchBucket"){

```



```
        echo "The specified bucket does not exist.";
    }
    throw $caught;
}
}
```

Python

SDK per Python (Boto3)

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Genera un URL prefirmato in grado di eseguire un'operazione S3 per un periodo di tempo limitato. Utilizza il pacchetto Requests per effettuare una richiesta con l'URL.

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)

def generate_presigned_url(s3_client, client_method, method_parameters,
    expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
```

```
:return: The presigned URL.
"""
try:
    url = s3_client.generate_presigned_url(
        ClientMethod=client_method, Params=method_parameters,
ExpiresIn=expires_in
    )
    logger.info("Got presigned URL: %s", url)
except ClientError:
    logger.exception(
        "Couldn't get a presigned URL for client method '%s'.", client_method
    )
    raise
return url

def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon S3 presigned URL demo.")
    print("-" * 88)

    parser = argparse.ArgumentParser()
    parser.add_argument("bucket", help="The name of the bucket.")
    parser.add_argument(
        "key",
        help="For a GET operation, the key of the object in Amazon S3. For a "
        "PUT operation, the name of a file to upload.",
    )
    parser.add_argument("action", choices=("get", "put"), help="The action to perform.")
    args = parser.parse_args()

    s3_client = boto3.client("s3")
    client_action = "get_object" if args.action == "get" else "put_object"
    url = generate_presigned_url(
        s3_client, client_action, {"Bucket": args.bucket, "Key": args.key}, 1000
    )

    print("Using the Requests package to send a request to the URL.")
    response = None
    if args.action == "get":
        response = requests.get(url)
```

```

        if response.status_code == 200:
            with open(args.key.split("/")[-1], 'wb') as object_file:
                object_file.write(response.content)
    elif args.action == "put":
        print("Putting data to the URL.")
        try:
            with open(args.key, "rb") as object_file:
                object_text = object_file.read()
                response = requests.put(url, data=object_text)
        except FileNotFoundError:
            print(
                f"Couldn't find {args.key}. For a PUT operation, the key must be
the "
                f"name of a file that exists on your computer."
            )

        if response is not None:
            print(f"Status: {response.status_code}\nReason: {response.reason}")

        print("-" * 88)

if __name__ == "__main__":
    usage_demo()

```

Genera una richiesta POST prefirmata per caricare un file.

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def generate_presigned_post(self, object_key, expires_in):
        """

```

AWS

Generate a presigned Amazon S3 POST request to upload a file.
A presigned POST can be used for a limited time to let someone without an account upload a file to a bucket.

:param object_key: The object key to identify the uploaded object.
:param expires_in: The number of seconds the presigned POST is valid.
:return: A dictionary that contains the URL and form fields that contain required access data.

```
"""
try:
    response = self.bucket.meta.client.generate_presigned_post(
        Bucket=self.bucket.name, Key=object_key, ExpiresIn=expires_in
    )
    logger.info("Got presigned POST URL: %s", response["url"])
except ClientError:
    logger.exception(
        "Couldn't get a presigned POST URL for bucket '%s' and object '%s'",
        self.bucket.name,
        object_key,
    )
    raise
return response
```

Ruby

SDK per Ruby

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
require 'aws-sdk-s3'
require 'net/http'
```

```
# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)
  url = bucket.object(object_key).presigned_url(:put)
  puts "Created presigned URL: #{url}"
  URI(url)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's
  why: #{e.message}"
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-file.txt"
  object_content = "This is the content of my-file.txt."

  bucket = Aws::S3::Bucket.new(bucket_name)
  presigned_url = get_presigned_url(bucket, object_key)
  return unless presigned_url

  response = Net::HTTP.start(presigned_url.host) do |http|
    http.send_request('PUT', presigned_url.request_uri, object_content,
    'content_type' => '')
  end

  case response
  when Net::HTTPSuccess
    puts 'Content uploaded!'
  else
    puts response.value
  end
end

run_demo if $PROGRAM_NAME == __FILE__
```

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea richieste di prefirma per oggetti S3 GET.

```
/// Generate a URL for a presigned GET request.
async fn get_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<(), Box<dyn Error>> {
    let expires_in = Duration::from_secs(expires_in);
    let presigned_request = client
        .get_object()
        .bucket(bucket)
        .key(object)
        .presigned(PresigningConfig::expires_in(expires_in?))
        .await?;

    println!("Object URI: {}", presigned_request.uri());
    let valid_until = chrono::offset::Local::now() + expires_in;
    println!("Valid until: {valid_until}");

    Ok(())
}
```

Crea richieste di prefirma per oggetti S3 PUT.

```
async fn put_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
```

```

) -> Result<String, S3ExampleError> {
  let expires_in: std::time::Duration =
std::time::Duration::from_secs(expires_in);
  let expires_in: aws_sdk_s3::presigning::PresigningConfig =
    PresigningConfig::expires_in(expires_in).map_err(|err| {
      S3ExampleError::new(format!(
        "Failed to convert expiration to PresigningConfig: {err:?}")
      ))
    })?;
  let presigned_request = client
    .put_object()
    .bucket(bucket)
    .key(object)
    .presigned(expires_in)
    .await?;

  Ok(presigned_request.uri().into())
}

```

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea richieste prefirmate per oggetti S3 GET.

```

" iv_bucket_name is the bucket name
" iv_key is the object name like "myfile.txt"

DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

"Upload a nice Hello World file to an S3 bucket."
TRY.
  DATA(lv_contents) = cl_abap_codepage=>convert_to( 'Hello, World' ).
  lo_s3->putobject(

```

```

        iv_bucket = iv_bucket_name
        iv_key = iv_key
        iv_body = lv_contents
        iv_contenttype = 'text/plain' ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" now generate a presigned URL with a 600-second expiration
DATA(lo_presigner) = lo_s3->get_presigner( iv_expires_sec = 600 ).
" the presigner getobject() method has the same signature as
" lo_s3->getobject(), but it doesn't actually make the call.
" to the service. It just prepares a presigned URL for a future call
DATA(lo_presigned_req) = lo_presigner->getobject(
    iv_bucket = iv_bucket_name
    iv_key = iv_key ).

" You can provide this URL to a web page, user, email etc so they
" can retrieve the file. The URL will expire in 10 minutes.
ov_url = lo_presigned_req->get_url( ).

```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Creazione di un'applicazione di gestione delle risorse fotografiche che consente agli utenti di gestire le foto utilizzando etichette

L'esempio di codice seguente mostra come creare un'applicazione serverless che consente agli utenti di gestire le foto mediante etichette.

.NET

SDK per .NET

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su [GitHub](#).

Per approfondire l'origine di questo esempio, consulta il post su [AWS Community](#).

Servizi utilizzati in questo esempio

- Gateway API
- DynamoDB
- Lambda
- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SNS

C++

SDK per C++

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Per approfondire l'origine di questo esempio, consulta il post su [AWS Community](#).

Servizi utilizzati in questo esempio

- Gateway API
- DynamoDB
- Lambda
- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SNS

Java

SDK per Java 2.x

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Per approfondire l'origine di questo esempio, consulta il post su [AWS Community](#).

Servizi utilizzati in questo esempio

- Gateway API
- DynamoDB
- Lambda
- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SNS

JavaScript

SDK per JavaScript (v3)

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su. [GitHub](#)

Per approfondire l'origine di questo esempio, consulta il post su [AWS Community](#).

Servizi utilizzati in questo esempio

- Gateway API
- DynamoDB
- Lambda

- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SNS

Kotlin

SDK per Kotlin

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Per approfondire l'origine di questo esempio, consulta il post su [AWS Community](#).

Servizi utilizzati in questo esempio

- Gateway API
- DynamoDB
- Lambda
- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SNS

PHP

SDK per PHP

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Per approfondire l'origine di questo esempio, consulta il post su [AWS Community](#).

Servizi utilizzati in questo esempio

- Gateway API
- DynamoDB
- Lambda
- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SNS

Rust

SDK per Rust

Mostra come sviluppare un'applicazione per la gestione delle risorse fotografiche che rileva le etichette nelle immagini utilizzando Amazon Rekognition e le archivia per recuperarle in seguito.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Per approfondire l'origine di questo esempio, consulta il post su [AWS Community](#).

Servizi utilizzati in questo esempio

- Gateway API
- DynamoDB
- Lambda
- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SNS

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Una pagina Web che elenca gli oggetti Amazon S3 utilizzando un AWS SDK

Il codice di esempio seguente mostra come elencare gli oggetti Amazon S3 in una pagina web.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Il codice seguente è il componente React pertinente che effettua chiamate all' AWS SDK. Una versione eseguibile dell'applicazione contenente questo componente è disponibile al link precedente. GitHub

```
import { useEffect, useState } from "react";
import {
  ListObjectsCommand,
  type ListObjectsCommandOutput,
  S3Client,
} from "@aws-sdk/client-s3";
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";
import "./App.css";

function App() {
  const [objects, setObjects] = useState<
    Required<ListObjectsCommandOutput>["Contents"]
  >([]);

  useEffect(() => {
    const client = new S3Client({
      region: "us-east-1",
      // Unless you have a public bucket, you'll need access to a private bucket.
      // One way to do this is to create an Amazon Cognito identity pool, attach
      // a role to the pool,
      // and grant the role access to the 's3:GetObject' action.
      //
      // You'll also need to configure the CORS settings on the bucket to allow
      // traffic from
      // this example site. Here's an example configuration that allows all
      // origins. Don't
      // do this in production.
      //
```

```
// {
//   "AllowedHeaders": ["*"],
//   "AllowedMethods": ["GET"],
//   "AllowedOrigins": ["*"],
//   "ExposeHeaders": [],
// },
//]
//
credentials: fromCognitoIdentityPool({
  clientConfig: { region: "us-east-1" },
  identityPoolId: "<YOUR_IDENTITY_POOL_ID>",
}),
});
const command = new ListObjectsCommand({ Bucket: "bucket-name" });
client.send(command).then(({ Contents }) => setObjects(Contents || []));
}, []);

return (
  <div className="App">
    {objects.map((o) => (
      <div key={o.ETag}>{o.Key}</div>
    ))}
  </div>
);
}

export default App;
```

- Per i dettagli sull'API, consulta la sezione API [ListObjects](#) Reference AWS SDK per JavaScript .

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Creazione di un'applicazione Amazon Textract explorer

Gli esempi di codice seguenti mostrano come esplorare l'output di Amazon Textract tramite un'applicazione interattiva.

JavaScript

SDK per JavaScript (v3)

Mostra come utilizzare per AWS SDK per JavaScript creare un'applicazione React che utilizza Amazon Textract per estrarre dati dall'immagine di un documento e visualizzarli in una pagina Web interattiva. Questo esempio viene eseguito in un browser Web e richiede, come credenziali, un'identità autenticata Amazon Cognito. Utilizza Amazon Simple Storage Service (Amazon S3) per l'archiviazione e per le notifiche esegue il polling di una coda Amazon Simple Queue Service (Amazon SQS) sottoscritta a un argomento Amazon Simple Notification Service (Amazon SNS).

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- Amazon Cognito Identity
- Simple Storage Service (Amazon S3)
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK per Python (Boto3)

Mostra come utilizzarlo AWS SDK per Python (Boto3) con Amazon Textract per rilevare elementi di testo, moduli e tabelle nell'immagine di un documento. L'immagine di input e l'output di Amazon Textract sono mostrati in un'applicazione Tkinter che consente di esplorare gli elementi rilevati.

- Invia un'immagine del documento ad Amazon Textract ed esplora l'output degli elementi rilevati.
- Invia immagini direttamente ad Amazon Textract o tramite un bucket Amazon Simple Storage Service (Amazon S3).
- Utilizza le API asincrone per avviare un processo che pubblica una notifica in un argomento Amazon Simple Notification Service (Amazon SNS) al suo termine.

- Esegue il polling di una coda Amazon Simple Queue Service (Amazon SQS) per un messaggio di completamento del processo e visualizza i risultati.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- Amazon Cognito Identity
- Simple Storage Service (Amazon S3)
- Amazon SNS
- Amazon SQS
- Amazon Textract

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Elimina tutti gli oggetti in un determinato bucket Amazon S3 utilizzando un AWS SDK

L'esempio di codice seguente mostra come eliminare più oggetti da un bucket Amazon S3.

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina tutti gli oggetti in un bucket Amazon S3 specificato.

```
import {
  DeleteObjectsCommand,
  paginateListObjectsV2,
  S3Client,
} from "@aws-sdk/client-s3";
```



```
/**
 *
 * @param {{ bucketName: string }} config
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  try {
    console.log(`Deleting all objects in bucket: ${bucketName}`);

    const paginator = paginateListObjectsV2(
      { client },
      {
        Bucket: bucketName,
      },
    );

    const objectKeys = [];
    for await (const { Contents } of paginator) {
      objectKeys.push(...Contents.map((obj) => ({ Key: obj.Key })));
    }

    const deleteCommand = new DeleteObjectsCommand({
      Bucket: bucketName,
      Delete: { Objects: objectKeys },
    });

    await client.send(deleteCommand);

    console.log(`All objects deleted from bucket: ${bucketName}`);
  } catch (caught) {
    if (caught instanceof Error) {
      console.error(
        `Failed to empty ${bucketName}. ${caught.name}: ${caught.message}`,
      );
    }
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const options = {
```

```
    bucketName: {  
      type: "string",  
    },  
  },  
};  
  
const { values } = parseArgs({ options });  
main(values);  
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per JavaScript .
 - [DeleteObjects](#)
 - [ListObjectsV2](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Eliminare caricamenti multipart incompleti su Amazon S3 utilizzando un AWS SDK

L'esempio di codice seguente mostra come eliminare o arrestare i caricamenti in più parti incompleti di Amazon S3.

Java

SDK per Java 2.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Per arrestare i caricamenti in più parti in corso o incompleti per qualsiasi motivo, è possibile creare un elenco e poi eliminarli come mostrato nell'esempio seguente.

```
/**  
 * Aborts all incomplete multipart uploads from the specified S3 bucket.  
 */
```

```

    * @param bucketName the name of the S3 bucket
    */
    public static void abortIncompleteMultipartUploadsFromList(String bucketName)
    {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
        ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        ListMultipartUploadsResponse response =
        s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(upload.key())
                .expectedBucketOwner(accountId)
                .uploadId(upload.uploadId())
                .build();

            AbortMultipartUploadResponse abortMultipartUploadResponse =
            s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                logger.info("Upload ID [{}] to bucket [{}] successfully
                aborted.", upload.uploadId(), bucketName);
            }
        }
    }
}

```

Per eliminare i caricamenti in più parti incompleti avviati prima o dopo una data, è possibile eliminare selettivamente i caricamenti in più parti in base a un point-in-time specifico, come mostrato nell'esempio seguente.

```

/**
 * Aborts incomplete multipart uploads older than the specified point in
 * time.
 *
 * @param bucketName the name of the S3 bucket
 * @param pointInTime the cutoff time; uploads initiated before this are
 * aborted

```

```

    */
    static void abortIncompleteMultipartUploadsOlderThan(String bucketName,
Instant pointInTime) {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            logger.info("Found multipartUpload with upload ID [{}], initiated
[{}]", upload.uploadId(), upload.initiated());
            if (upload.initiated().isBefore(pointInTime)) {
                abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(upload.key())
                    .expectedBucketOwner(accountId)
                    .uploadId(upload.uploadId())
                    .build();

                AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
                if
(abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                    logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
                }
            }
        }
    }
}

```

Se hai accesso all'ID di caricamento dopo aver avviato un caricamento in più parti, puoi eliminare il caricamento in corso utilizzando l'ID.

```

/**
 * Aborts a multipart upload using the upload ID.
 *

```

```

    * @param bucketName the name of the S3 bucket
    * @param key          the object key
    */
    static void abortMultipartUploadUsingUploadId(String bucketName, String key)
    {
        String uploadId = startUploadReturningUploadId(bucketName, key);
        AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b
            .uploadId(uploadId)
            .bucket(bucketName)
            .key(key));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
        }
    }
}

```

Per eliminare sistematicamente i caricamenti in più parti incompleti più vecchi di un determinato numero di giorni, imposta una configurazione del ciclo di vita per il bucket. L'esempio seguente mostra come creare una regola per eliminare i caricamenti incompleti con più di 7 giorni.

```

/**
 * Configures a lifecycle rule to abort incomplete multipart uploads after 7
days.
 *
 * @param bucketName the name of the S3 bucket
 */
    static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
        Collection<LifecycleRule> lifeCycleRules =
List.of(LifecycleRule.builder()
            .abortIncompleteMultipartUpload(b -> b.
                daysAfterInitiation(7))
            .status("Enabled")
            .filter(SdkBuilder::build) // Filter element is required.
            .build());

        PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
            .bucket(bucketName)
            .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));
    }
}

```

```
        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
        } else {
            logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK for Java 2.x .
 - [AbortMultipartUpload](#)
 - [ListMultipartUploads](#)
 - [PutBucketLifecycleConfiguration](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Rileva i DPI nelle immagini con Amazon Rekognition utilizzando un AWS SDK

L'esempio di codice seguente mostra come creare un'applicazione che utilizza Amazon Rekognition per rilevare dispositivi di protezione individuale (DPI) nelle immagini.

Java

SDK per Java 2.x

Mostra come creare una AWS Lambda funzione che rileva le immagini con dispositivi di protezione individuale.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- DynamoDB
- Amazon Rekognition

- Simple Storage Service (Amazon S3)
- Amazon SES

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Rileva le entità nel testo estratto da un'immagine utilizzando un AWS SDK

L'esempio di codice seguente mostra come utilizzare Amazon Comprehend per rilevare le entità nel testo estratto da Amazon Textract da un'immagine archiviata in Amazon S3.

Python

SDK per Python (Boto3)

Mostra come utilizzarlo AWS SDK per Python (Boto3) in un notebook Jupyter per rilevare entità nel testo estratto da un'immagine. In questo esempio viene utilizzato Amazon Textract per estrarre il testo da un'immagine archiviata in Amazon Simple Storage Service (Amazon S3) e Amazon Comprehend per rilevare le entità nel testo estratto.

Questo esempio è un notebook Jupyter e deve essere eseguito in un ambiente in grado di ospitare notebook. Per istruzioni su come eseguire l'esempio utilizzando Amazon SageMaker AI, consulta le istruzioni riportate in [TextractAndComprehendNotebook.ipynb](#).

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- Amazon Comprehend
- Simple Storage Service (Amazon S3)
- Amazon Textract

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Rileva i volti in un'immagine utilizzando un AWS SDK

L'esempio di codice seguente mostra come:

- Salva un'immagine in un bucket Amazon S3.
- Utilizza Amazon Rekognition per rilevare i dettagli del viso, come fascia di età, sesso ed emozione (ad esempio sorridente).
- Visualizzare questi dettagli.

Rust

SDK per Rust

Salva l'immagine in un bucket Amazon S3 con il prefisso uploads, utilizza Amazon Rekognition per rilevare i dettagli del viso, come fascia di età, sesso ed emozione (sorridente, ecc.) e visualizza tali dettagli.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Simple Storage Service (Amazon S3)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Rileva oggetti nelle immagini con Amazon Rekognition utilizzando un AWS SDK

Gli esempi di codice seguenti mostrano come creare un'applicazione che utilizza Amazon Rekognition per rilevare oggetti in base a una categoria nelle immagini.

.NET

SDK per .NET

Mostra come utilizzare l'API .NET di Amazon Rekognition per creare un'applicazione che utilizza Amazon Rekognition per identificare gli oggetti in base a una categoria nelle immagini

situate in un bucket Amazon Simple Storage Service (Amazon S3). L'applicazione invia all'amministratore una notifica e-mail sui risultati tramite Amazon Simple Email Service (Amazon SES).

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SES

Java

SDK per Java 2.x

Mostra come utilizzare l'API Java di Amazon Rekognition per creare un'applicazione che utilizza Amazon Rekognition per identificare gli oggetti in base a una categoria nelle immagini situate in un bucket Amazon Simple Storage Service (Amazon S3). L'applicazione invia all'amministratore una notifica e-mail sui risultati tramite Amazon Simple Email Service (Amazon SES).

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SES

JavaScript

SDK per JavaScript (v3)

Mostra come usare Amazon Rekognition AWS SDK per JavaScript con la per creare un'app che utilizzi Amazon Rekognition per identificare gli oggetti per categoria nelle immagini

che si trovano in un bucket Amazon Simple Storage Service (Amazon S3). L'applicazione invia all'amministratore una notifica e-mail sui risultati tramite Amazon Simple Email Service (Amazon SES).

Scopri come:

- Creare un utente non autenticato tramite Amazon Cognito.
- Analizza le immagini per rilevare gli oggetti tramite Amazon Rekognition.
- Verificare un indirizzo e-mail per Amazon SES.
- Inviare una notifica e-mail tramite Amazon SES.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su [GitHub](#)

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SES

Kotlin

SDK per Kotlin

Mostra come utilizzare l'API Kotlin di Amazon Rekognition per creare un'applicazione che utilizza Amazon Rekognition per identificare gli oggetti in base a una categoria nelle immagini situate in un bucket Amazon Simple Storage Service (Amazon S3). L'applicazione invia all'amministratore una notifica e-mail sui risultati tramite Amazon Simple Email Service (Amazon SES).

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SES

Python

SDK per Python (Boto3)

Illustra come utilizzare il AWS SDK per Python (Boto3) per creare un'applicazione Web che consenta di eseguire le seguenti operazioni:

- Caricamento di foto in un bucket Amazon Simple Storage Service (Amazon S3).
- Utilizzo di Amazon Rekognition per analizzare ed etichettare le foto.
- Utilizzo di Amazon Simple Email Service (Amazon SES) per inviare report dell'analisi delle immagini tramite e-mail.

Questo esempio contiene due componenti principali: una pagina web scritta in JavaScript che è costruita con React e un servizio REST scritto in Python che è costruito con Flask-RESTful

È possibile utilizzare la pagina web React per:

- Visualizzare un elenco di immagini archiviate nel bucket S3.
- Caricare le immagini dal computer nel bucket S3.
- Visualizzare immagini ed etichette che identificano gli elementi rilevati nell'immagine.
- Ottenere un report relativo a tutte le immagini nel bucket S3 e inviarlo tramite email.

La pagina web richiama il servizio REST. Il servizio invia richieste a AWS per eseguire le seguenti operazioni:

- Ottenere e filtrare l'elenco delle immagini nel bucket S3.
- Caricare le foto nel bucket S3.
- Utilizzare Amazon Rekognition per analizzare le singole foto e ottenere un elenco di etichette che identificano gli articoli rilevati al loro interno.
- Analizzare tutte le foto presenti nel bucket S3 e usare Amazon SES per inviare un report tramite e-mail.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#)

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SES

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Rileva persone e oggetti in un video con Amazon Rekognition utilizzando un AWS SDK

Gli esempi di codice seguenti mostrano come rilevare persone e oggetti in un video con Amazon Rekognition.

Java

SDK per Java 2.x

Mostra come utilizzare l'API Java di Amazon Rekognition per creare un'applicazione che rileva volti e oggetti nei video situati in un bucket Amazon Simple Storage Service (Amazon S3). L'applicazione invia all'amministratore una notifica e-mail sui risultati tramite Amazon Simple Email Service (Amazon SES).

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK per Python (Boto3)

Usa Amazon Rekognition per rilevare volti, oggetti e persone nei video avviando processi di rilevamento asincrono. Questo esempio, inoltre, configura Amazon Rekognition per notificare un argomento Amazon Simple Notification Service (Amazon SNS) al completamento dei processi e sottoscrive una coda Amazon Simple Queue Service (Amazon SQS) all'argomento. Quando la coda riceve un messaggio su un processo, questo viene recuperato e vengono restituiti i risultati.

Questo esempio è visualizzato al meglio su GitHub. Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, vedi l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- Amazon Rekognition
- Simple Storage Service (Amazon S3)
- Amazon SES
- Amazon SNS
- Amazon SQS

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Scaricare l'oggetto S3 "directory" da un bucket Amazon Simple Storage Service (Amazon S3)

L'esempio di codice seguente mostra come scaricare e filtrare il contenuto dell'oggetto "directories" del bucket Amazon S3.

Java

SDK per Java 2.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Questo esempio mostra come utilizzare le «directory» [S3TransferManager](#) in AWS SDK for Java 2.x per scaricare le «directory» da un bucket Amazon S3. Viene inoltre illustrato come [DownloadFilters](#) utilizzarlo nella richiesta.

```
/**
 * For standard buckets, S3 provides the illusion of a directory structure
 * through the use of keys. When you upload
```

```

    * an object to an S3 bucket, you specify a key, which is essentially the
    "path" to the object. The key can contain
    * forward slashes ("/") to make it appear as if the object is stored in a
    directory structure, but this is just a
    * logical representation, not an actual directory.
    * <p><pre>
    * In this example, our S3 bucket contains the following objects:
    *
    * folder1/file1.txt
    * folder1/file2.txt
    * folder1/file3.txt
    * folder2/file1.txt
    * folder2/file2.txt
    * folder2/file3.txt
    * folder3/file1.txt
    * folder3/file2.txt
    * folder3/file3.txt
    *
    * When method `downloadS3Directories` is invoked with
    * `destinationPathURI` set to `/test`, the downloaded
    * directory looks like:
    *
    * |- test
    *   |- folder1
    *     |- file1.txt
    *     |- file2.txt
    *     |- file3.txt
    *   |- folder3
    *     |- file1.txt
    *     |- file2.txt
    *     |- file3.txt
    * </pre>
    *
    * @param transferManager    An S3TransferManager instance.
    * @param destinationPathURI local directory to hold the downloaded S3
    'directories' and files.
    * @param bucketName         The S3 bucket that contains the 'directories' to
    download.
    * @return The number of objects (files, in this case) that were downloaded.
    */
    public Integer downloadS3Directories(S3TransferManager transferManager,
                                         URI destinationPathURI, String
    bucketName) {

```

```

        // Define the filters for which 'directories' we want to download.
        DownloadFilter folder1Filter = (S3Object s3object) ->
s3object.key().startsWith("folder1/");
        DownloadFilter folder3Filter = (S3Object s3object) ->
s3object.key().startsWith("folder3/");
        DownloadFilter folderFilter = s3object ->
folder1Filter.or(folder3Filter).test(s3object);

        DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                .destination(Paths.get(destinationPathURI))
                .bucket(bucketName)
                .filter(folderFilter)
                .build());
        CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

        Integer numFilesInFolder1 =
Paths.get(destinationPathURI).resolve("folder1").toFile().list().length;
        Integer numFilesInFolder3 =
Paths.get(destinationPathURI).resolve("folder3").toFile().list().length;

        try {
            assert numFilesInFolder1 == 3;
            assert numFilesInFolder3 == 3;
            assert !
Paths.get(destinationPathURI).resolve("folder2").toFile().exists(); // `folder2`
was not downloaded.
        } catch (AssertionError e) {
            logger.error("An assertion failed.");
        }

        completedDirectoryDownload.failedTransfers()
                .forEach(fail -> logger.warn("Object failed to transfer [{})",
fail.exception().getMessage()));
        return numFilesInFolder1 + numFilesInFolder3;
    }

```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Scaricare tutti gli oggetti da un bucket Amazon Simple Storage Service (Amazon S3) in una directory locale

L'esempio di codice seguente mostra come scaricare tutti gli oggetti da un bucket Amazon Simple Storage Service (Amazon S3) in una directory locale.

Java

SDK per Java 2.x

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Usa un [S3TransferManager](#) per [scaricare tutti gli oggetti S3](#) nello stesso bucket S3. Visualizza il [file completo](#) ed esegui il [test](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;

    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
        URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
            transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
```



```
        .destination(Paths.get(destinationPathURI))
        .bucket(bucketName)
        .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}
```

- Per i dettagli sull'API, consulta la sezione API [DownloadDirectory](#) Reference AWS SDK for Java 2.x .

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Scarica uno stream di dimensioni sconosciute da un oggetto Amazon S3 utilizzando un AWS SDK

Gli esempi di codice seguenti mostrano come scaricare un flusso di dimensioni sconosciute in un oggetto Amazon S3.

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import ArgumentParser
import AWSClientRuntime
import AWSS3
import Foundation
```

```
import Smithy
import SmithyHTTPAPI
import SmithyStreams

/// Download a file from the specified bucket.
///
/// - Parameters:
///   - bucket: The Amazon S3 bucket name to get the file from.
///   - key: The name (or path) of the file to download from the bucket.
///   - destPath: The pathname on the local filesystem at which to store
///     the downloaded file.
func downloadFile(bucket: String, key: String, destPath: String?) async
throws {
    let fileURL: URL

    // If no destination path was provided, use the key as the name to use
    // for the file in the downloads folder.

    if destPath == nil {
        do {
            try fileURL = FileManager.default.url(
                for: .downloadsDirectory,
                in: .userDomainMask,
                appropriateFor: URL(string: key),
                create: true
            ).appendingPathComponent(key)
        } catch {
            throw TransferError.directoryError
        }
    } else {
        fileURL = URL(fileURLWithPath: destPath!)
    }

    let config = try await S3Client.S3ClientConfiguration(region: region)
    let s3Client = S3Client(config: config)

    // Create a `FileHandle` referencing the local destination. Then
    // create a `ByteStream` from that.

    FileManager.default.createFile(atPath: fileURL.path, contents: nil,
    attributes: nil)
    let fileHandle = try FileHandle(forWritingTo: fileURL)
```

```
// Download the file using `GetObject`.

let getInput = GetObjectInput(
    bucket: bucket,
    key: key
)

do {
    let getOutput = try await s3Client.getObject(input: getInput)

    guard let body = getOutput.body else {
download")      throw TransferError.downloadError("Error: No data returned for
    }

    // If the body is returned as a `Data` object, write that to the
    // file. If it's a stream, read the stream chunk by chunk,
    // appending each chunk to the destination file.

    switch body {
    case .data:
        guard let data = try await body.readData() else {
            throw TransferError.downloadError("Download error")
        }

        // Write the `Data` to the file.

        do {
            try data.write(to: fileURL)
        } catch {
            throw TransferError.writeError
        }
        break

    case .stream(let stream as ReadableStream):
        while (true) {
            let chunk = try await stream.readAsync(upToCount: 5 * 1024 *
1024)

            guard let chunk = chunk else {
                break
            }

            // Write the chunk to the destination file.
```

```
        do {
            try fileHandle.write(contentsOf: chunk)
        } catch {
            throw TransferError.writeError
        }
    }

    break
default:
    throw TransferError.downloadError("Received data is unknown
object type")
}
} catch {
    throw TransferError.downloadError("Error downloading the file:
\\(error)")
}

print("File downloaded to \\(fileURL.path).")
}
```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Ottieni un oggetto Amazon S3 da un punto di Multi-Region accesso utilizzando un AWS SDK

Il seguente esempio di codice mostra come ottenere un oggetto da un Multi-Region Access Point.

Kotlin

SDK per Kotlin

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Configura il client S3 per l'utilizzo dell'algoritmo di firma Asymmetric Sigv4 (Sigv4a).

```
suspend fun createS3Client(): S3Client {  
    // Configure your S3Client to use the Asymmetric SigV4 (SigV4a)  
    signing algorithm.  
    val sigV4aScheme = SigV4AsymmetricAuthScheme(DefaultAwsSigner)  
    val s3 = S3Client.fromEnvironment {  
        authSchemes = listOf(sigV4aScheme)  
    }  
    return s3  
}
```

Utilizzate l'ARN del punto di Multi-Region accesso anziché il nome del bucket per recuperare l'oggetto.

```
suspend fun getObjectFromMrap(  
    s3: S3Client,  
    mrapArn: String,  
    keyName: String,  
): String? {  
    val request = GetObjectRequest {  
        bucket = mrapArn // Use the ARN instead of the bucket name for object  
        operations.  
        key = keyName  
    }  
  
    var stringObj: String? = null  
    s3.getObject(request) { resp ->  
        stringObj = resp.body?.decodeToString()  
        if (stringObj != null) {  
            println("Successfully read $keyName from $mrapArn")  
        }  
    }  
    return stringObj  
}
```

- Per ulteriori informazioni, consulta la [Guida per gli sviluppatori di AWS SDK per Swift](#).
- Per i dettagli sull'API, consulta il riferimento [GetObject](#) all'API AWS SDK for Kotlin.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Ottieni un oggetto da un bucket Amazon S3 utilizzando un AWS SDK, specificando un'intestazione If-Modified-Since

L'esempio di codice seguente mostra come leggere i dati da un oggetto in un bucket S3, ma solo se il bucket non è stato modificato dall'ultimo recupero.

Rust

SDK per Rust

Note

C'è altro da fare. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
use aws_sdk_s3::{
    error::SdkError,
    primitives::{ByteStream, DateTime, DateTimeFormat},
    Client,
};
use s3_code_examples::error::S3ExampleError;
use tracing::{error, warn};

const KEY: &str = "key";
const BODY: &str = "Hello, world!";

/// Demonstrate how `if-modified-since` reports that matching objects haven't
/// changed.
///
/// # Steps
/// - Create a bucket.
/// - Put an object in the bucket.
/// - Get the bucket headers.
/// - Get the bucket headers again but only if modified.
/// - Delete the bucket.
#[tokio::main]
async fn main() -> Result<(), S3ExampleError> {
```

```
tracing_subscriber::fmt::init();

// Get a new UUID to use when creating a unique bucket name.
let uuid = uuid::Uuid::new_v4();

// Load the AWS configuration from the environment.
let client = Client::new(&aws_config::load_from_env().await);

// Generate a unique bucket name using the previously generated UUID.
// Then create a new bucket with that name.
let bucket_name = format!("if-modified-since-{{uuid}}");
client
    .create_bucket()
    .bucket(bucket_name.clone())
    .send()
    .await?;

// Create a new object in the bucket whose name is `KEY` and whose
// contents are `BODY`.
let put_object_output = client
    .put_object()
    .bucket(bucket_name.as_str())
    .key(KEY)
    .body(ByteStream::from_static(BODY.as_bytes()))
    .send()
    .await;

// If the `PutObject` succeeded, get the eTag string from it. Otherwise,
// report an error and return an empty string.
let e_tag_1 = match put_object_output {
    Ok(put_object) => put_object.e_tag.unwrap(),
    Err(err) => {
        error!("{err:?}");
        String::new()
    }
};

// Request the object's headers.
let head_object_output = client
    .head_object()
    .bucket(bucket_name.as_str())
    .key(KEY)
    .send()
    .await;
```

```
// If the `HeadObject` request succeeded, create a tuple containing the
// values of the headers `last-modified` and `etag`. If the request
// failed, return the error in a tuple instead.
let (last_modified, e_tag_2) = match head_object_output {
    Ok(head_object) => (
        Ok(head_object.last_modified().cloned().unwrap()),
        head_object.e_tag.unwrap(),
    ),
    Err(err) => (Err(err), String::new()),
};

warn!("last modified: {last_modified:?}");
assert_eq!(
    e_tag_1, e_tag_2,
    "PutObject and first GetObject had differing eTags"
);

println!("First value of last_modified: {last_modified:?}");
println!("First tag: {}\\n", e_tag_1);

// Send a second `HeadObject` request. This time, the `if_modified_since`
// option is specified, giving the `last_modified` value returned by the
// first call to `HeadObject`.
//
// Since the object hasn't been changed, and there are no other objects in
// the bucket, there should be no matching objects.

let head_object_output = client
    .head_object()
    .bucket(bucket_name.as_str())
    .key(KEY)
    .if_modified_since(last_modified.unwrap())
    .send()
    .await;

// If the `HeadObject` request succeeded, the result is a tuple containing
// the `last_modified` and `e_tag_1` properties. This is _not_ the expected
// result.
//
// The _expected_ result of the second call to `HeadObject` is an
// `SdkError::ServiceError` containing the HTTP error response. If that's
// the case and the HTTP status is 304 (not modified), the output is a
// tuple containing the values of the HTTP `last-modified` and `etag`
```



```

// headers.
//
// If any other HTTP error occurred, the error is returned as an
// `SdkError::ServiceError`.

let (last_modified, e_tag_2) = match head_object_output {
    Ok(head_object) => (
        Ok(head_object.last_modified().cloned().unwrap()),
        head_object.e_tag.unwrap(),
    ),
    Err(err) => match err {
        SdkError::ServiceError(err) => {
            // Get the raw HTTP response. If its status is 304, the
            // object has not changed. This is the expected code path.
            let http = err.raw();
            match http.status().as_u16() {
                // If the HTTP status is 304: Not Modified, return a
                // tuple containing the values of the HTTP
                // `last-modified` and `etag` headers.
                304 => (
                    Ok(DateTime::from_str(
                        http.headers().get("last-modified").unwrap(),
                        DateTimeFormat::HttpDate,
                    )
                    .unwrap()),
                    http.headers().get("etag").map(|t| t.into()).unwrap(),
                ),
                // Any other HTTP status code is returned as an
                // `SdkError::ServiceError`.
                _ => (Err(SdkError::ServiceError(err)), String::new()),
            }
        }
        // Any other kind of error is returned in a tuple containing the
        // error and an empty string.
        _ => (Err(err), String::new()),
    },
};

warn!("last modified: {last_modified:?}");
assert_eq!(
    e_tag_1, e_tag_2,
    "PutObject and second HeadObject had different eTags"
);

```

```
println!("Second value of last modified: {last_modified:?}");
println!("Second tag: {}", e_tag_2);

// Clean up by deleting the object and the bucket.
client
    .delete_object()
    .bucket(bucket_name.as_str())
    .key(KEY)
    .send()
    .await?;

client
    .delete_bucket()
    .bucket(bucket_name.as_str())
    .send()
    .await?;

Ok(())
}
```

- Per i dettagli sulle API, consulta la [GetObject](#) guida di riferimento all'API AWS SDK for Rust.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Inizia a utilizzare la crittografia per oggetti Amazon S3 utilizzando un AWS SDK

L'esempio di codice seguente mostra come iniziare a utilizzare la crittografia per gli oggetti Amazon S3.

.NET

SDK per .NET

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to apply client encryption to an object in an
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class SSEClientEncryption
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "exampleobject.txt";
        string copyTargetKeyName = "examplecopy.txt";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2.
        IAmazonS3 client = new AmazonS3Client();

        try
        {
            // Create an encryption key.
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);

            // Upload the object.
            PutObjectRequest putObjectRequest = await
UploadObjectAsync(client, bucketName, keyName, base64Key);

            // Download the object and verify that its contents match what
            you uploaded.
            await DownloadObjectAsync(client, bucketName, keyName, base64Key,
putObjectRequest);
        }
    }
}
```

```

        // Get object metadata and verify that the object uses AES-256
encryption.
        await GetObjectMetadataAsync(client, bucketName, keyName,
base64Key);

        // Copy both the source and target objects using server-side
encryption with
        // an encryption key.
        await CopyObjectAsync(client, bucketName, keyName,
copyTargetKeyName, aesEncryption, base64Key);
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error: {ex.Message}");
    }
}

/// <summary>
/// Uploads an object to an Amazon S3 bucket.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
call
/// PutObjectAsync.</param>
/// <param name="bucketName">The name of the Amazon S3 bucket to which
the
/// object will be uploaded.</param>
/// <param name="keyName">The name of the object to upload to the Amazon
S3
/// bucket.</param>
/// <param name="base64Key">The encryption key.</param>
/// <returns>The PutObjectRequest object for use by
DownloadObjectAsync.</returns>
public static async Task<PutObjectRequest> UploadObjectAsync(
    IAmazonS3 client,
    string bucketName,
    string keyName,
    string base64Key)
{
    PutObjectRequest putObjectRequest = new PutObjectRequest
    {
        BucketName = bucketName,
        Key = keyName,
        ContentBody = "sample text",
    }
}

```

```

        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
        ServerSideEncryptionCustomerProvidedKey = base64Key,
    };
    PutObjectResponse putObjectResponse = await
client.PutObjectAsync(putObjectRequest);
    return putObjectRequest;
}

/// <summary>
/// Downloads an encrypted object from an Amazon S3 bucket.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
call
/// GetObjectAsync.</param>
/// <param name="bucketName">The name of the Amazon S3 bucket where the
object
/// is located.</param>
/// <param name="keyName">The name of the Amazon S3 object to download.</
param>
/// <param name="base64Key">The encryption key used to encrypt the
/// object.</param>
/// <param name="putObjectRequest">The PutObjectRequest used to upload
/// the object.</param>
public static async Task DownloadObjectAsync(
    IAmazonS3 client,
    string bucketName,
    string keyName,
    string base64Key,
    PutObjectRequest putObjectRequest)
{
    GetObjectRequest getObjectRequest = new GetObjectRequest
    {
        BucketName = bucketName,
        Key = keyName,

        // Provide encryption information for the object stored in Amazon
S3.

        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
        ServerSideEncryptionCustomerProvidedKey = base64Key,
    };

```

```

        using (GetObjectResponse getResponse = await
client.GetObjectAsync(getObjectRequest))
        using (StreamReader reader = new
StreamReader(getResponse.ResponseStream))
        {
            string content = reader.ReadToEnd();
            if (string.Compare(putObjectRequest.ContentBody, content) == 0)
            {
                Console.WriteLine("Object content is same as we uploaded");
            }
            else
            {
                Console.WriteLine("Error...Object content is not same.");
            }

            if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
            {
                Console.WriteLine("Object encryption method is AES256, same
as we set");
            }
            else
            {
                Console.WriteLine("Error...Object encryption method is not
the same as AES256 we set");
            }
        }
    }

    /// <summary>
    /// Retrieves the metadata associated with an Amazon S3 object.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used
    /// to call GetObjectMetadataAsync.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket containing
the
    /// object for which we want to retrieve metadata.</param>
    /// <param name="keyName">The name of the object for which we wish to
    /// retrieve the metadata.</param>
    /// <param name="base64Key">The encryption key associated with the
    /// object.</param>
    public static async Task GetObjectMetadataAsync(
        IAmazonS3 client,
        string bucketName,

```

```

        string keyName,
        string base64Key)
    {
        GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
        {
            BucketName = bucketName,
            Key = keyName,

            // The object stored in Amazon S3 is encrypted, so provide the
necessary encryption information.
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = base64Key,
        };

        GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
        Console.WriteLine("The object metadata show encryption method used
is: {0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
    }

    /// <summary>
    /// Copies an encrypted object from one Amazon S3 bucket to another.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// CopyObjectAsync.</param>
    /// <param name="bucketName">The Amazon S3 bucket containing the object
    /// to copy.</param>
    /// <param name="keyName">The name of the object to copy.</param>
    /// <param name="copyTargetKeyName">The Amazon S3 bucket to which the
object
    /// will be copied.</param>
    /// <param name="aesEncryption">The encryption type to use.</param>
    /// <param name="base64Key">The encryption key to use.</param>
    public static async Task CopyObjectAsync(
        IAmazonS3 client,
        string bucketName,
        string keyName,
        string copyTargetKeyName,
        Aes aesEncryption,
        string base64Key)
    {

```

```
        aesEncryption.GenerateKey();
        string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

        CopyObjectRequest copyRequest = new CopyObjectRequest
        {
            SourceBucket = bucketName,
            SourceKey = keyName,
            DestinationBucket = bucketName,
            DestinationKey = copyTargetKeyName,

            // Information about the source object's encryption.
            CopySourceServerSideEncryptionCustomerMethod =
            ServerSideEncryptionCustomerMethod.AES256,
            CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,

            // Information about the target object's encryption.
            ServerSideEncryptionCustomerMethod =
            ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = copyBase64Key,
        };
        await client.CopyObjectAsync(copyRequest);
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per .NET .
 - [CopyObject](#)
 - [GetObject](#)
 - [GetObjectMetadata](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Inizia a usare i tag per gli oggetti Amazon S3 utilizzando un AWS SDK

L'esempio di codice seguente mostra come utilizzare i tag con gli oggetti Amazon S3.

.NET

SDK per .NET

Note

C'è altro su GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to work with tags in Amazon Simple Storage
/// Service (Amazon S3) objects.
/// </summary>
public class ObjectTag
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "newobject.txt";
        string filePath = @"""" file path """";

        // Specify your bucket region (an example region is shown).
        RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

        var client = new AmazonS3Client(bucketRegion);
        await PutObjectsWithTagsAsync(client, bucketName, keyName, filePath);
    }

    /// <summary>
    /// This method uploads an object with tags. It then shows the tag
    /// values, changes the tags, and shows the new tags.
    /// </summary>
    /// <param name="client">The Initialized Amazon S3 client object used
    /// to call the methods to create and change an objects tags.</param>
```

```
/// <param name="bucketName">A string representing the name of the
/// bucket where the object will be stored.</param>
/// <param name="keyName">A string representing the key name of the
/// object to be tagged.</param>
/// <param name="filePath">The directory location and file name of the
/// object to be uploaded to the Amazon S3 bucket.</param>
public static async Task PutObjectsWithTagsAsync(IAmazonS3 client, string
bucketName, string keyName, string filePath)
{
    try
    {
        // Create an object with tags.
        var putRequest = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = keyName,
            FilePath = filePath,
            TagSet = new List<Tag>
            {
                new Tag { Key = "Keyx1", Value = "Value1" },
                new Tag { Key = "Keyx2", Value = "Value2" },
            },
        };

        PutObjectResponse response = await
client.PutObjectAsync(putRequest);

        // Now retrieve the new object's tags.
        GetObjectTaggingRequest getTagsRequest = new
GetObjectTaggingRequest()
        {
            BucketName = bucketName,
            Key = keyName,
        };

        GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);

        // Display the tag values.
        objectTags.Tagging
            .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));

        Tagging newTagSet = new Tagging()
```

```
        {
            TagSet = new List<Tag>
            {
                new Tag { Key = "Key3", Value = "Value3" },
                new Tag { Key = "Key4", Value = "Value4" },
            },
        };

        PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
        {
            BucketName = bucketName,
            Key = keyName,
            Tagging = newTagSet,
        };

        PutObjectTaggingResponse response2 = await
client.PutObjectTaggingAsync(putObjTagsRequest);

        // Retrieve the tags again and show the values.
        GetObjectTaggingRequest getTagsRequest2 = new
GetObjectTaggingRequest()
        {
            BucketName = bucketName,
            Key = keyName,
        };
        GetObjectTaggingResponse objectTags2 = await
client.GetObjectTaggingAsync(getTagsRequest2);

        objectTags2.Tagging
            .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine(
            $"Error: '{ex.Message}'");
    }
}
```

- Per i dettagli sull'API, consulta la [GetObjectTagging](#) sezione AWS SDK per .NET API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Guida introduttiva ai cluster di elaborazione dei Big Data

L'esempio di codice seguente mostra come:

- Crea una coppia di key pair EC2
- Configura lo storage e prepara l'applicazione
- Eseguire la pulizia delle risorse

Bash

AWS CLI con lo script Bash

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel [repository dei tutorial sugli esempi di codice per gli sviluppatori](#).

```
#!/bin/bash

# EMR Getting Started Tutorial Script
# This script automates the steps in the Amazon EMR Getting Started tutorial

set -euo pipefail

# Security: Set strict mode and trap errors
trap 'handle_error "Script interrupted or command failed"' ERR

# Set up logging with secure permissions
LOG_FILE="emr-tutorial.log"
touch "$LOG_FILE"
```

```
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EMR Getting Started Tutorial Script"
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "${BUCKET_NAME:-}" ]; then echo "- S3 Bucket: $BUCKET_NAME"; fi
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- EMR Cluster: $CLUSTER_ID"; fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo ""
    echo "=====
    echo "CLEANUP IN PROGRESS"
    echo "=====
    echo "Starting cleanup process..."

    # Terminate EMR cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Terminating EMR cluster: $CLUSTER_ID"
        aws emr terminate-clusters --cluster-ids "$CLUSTER_ID" 2>/dev/null ||
true

        echo "Waiting for cluster to terminate..."
        aws emr wait cluster-terminated --cluster-id "$CLUSTER_ID" 2>/dev/null ||
true
        echo "Cluster terminated successfully."
    fi

    # Delete S3 bucket and contents if it exists and is not shared
    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" != "true" ];
then
        echo "Deleting S3 bucket contents: $BUCKET_NAME"
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || true
    fi
}
```

```
        echo "Deleting S3 bucket: $BUCKET_NAME"
        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null || true
    fi

    # Remove temporary key pair file if created by this script
    if [ -f "${KEY_NAME_FILE:-}" ]; then
        rm -f "$KEY_NAME_FILE"
        echo "Removed temporary key pair file."
    fi

    echo "Cleanup completed."
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed"
fi

# Test AWS credentials
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
    text 2>/dev/null || true)

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="emr-${RANDOM_ID}"
fi
echo "Using bucket name: $BUCKET_NAME"

# Create S3 bucket with security best practices
echo "Creating S3 bucket: $BUCKET_NAME"
```

```
aws s3 mb "s3://$BUCKET_NAME" --region "${AWS_REGION:-us-east-1}" || handle_error
"Failed to create S3 bucket"

# Tag the bucket
aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=emr-
gs}]'

# Enable bucket versioning for safety
aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled || true

# Block public access to bucket
aws s3api put-public-access-block --bucket "$BUCKET_NAME" \
    --public-access-block-configuration \

"BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
|| true

# Enable encryption on bucket
aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' || true

echo "S3 bucket created successfully with security best practices."

# Create PySpark script
echo "Creating PySpark script: health_violations.py"
cat > health_violations.py << 'EOL'
import argparse

from pyspark.sql import SparkSession

def calculate_red_violations(data_source, output_uri):
    """
    Processes sample food establishment inspection data and queries the data to
    find the top 10 establishments
    with the most Red violations from 2006 to 2020.
```

```

        :param data_source: The URI of your food establishment data CSV, such as
        's3://emr-tutorial-bucket/food-establishment-data.csv'.
        :param output_uri: The URI where output is written, such as 's3://emr-
        tutorial-bucket/restaurant_violation_results'.
        """

        with SparkSession.builder.appName("Calculate Red Health
        Violations").getOrCreate() as spark:
            # Load the restaurant violation CSV data
            if data_source is not None:
                restaurants_df = spark.read.option("header", "true").csv(data_source)

            # Create an in-memory DataFrame to query
            restaurants_df.createOrReplaceTempView("restaurant_violations")

            # Create a DataFrame of the top 10 restaurants with the most Red
            violations
            top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
            total_red_violations
            FROM restaurant_violations
            WHERE violation_type = 'RED'
            GROUP BY name
            ORDER BY total_red_violations DESC LIMIT 10""")

            # Write the results to the specified output URI
            top_red_violation_restaurants.write.option("header",
            "true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3
        bucket location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
        location.")
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
EOL

# Secure the script file
chmod 600 health_violations.py

# Upload PySpark script to S3

```



```
echo "Uploading PySpark script to S3"
aws s3 cp health_violations.py "s3://$BUCKET_NAME/" --sse AES256 || handle_error
    "Failed to upload PySpark script"
echo "PySpark script uploaded successfully."

# Download and prepare sample data
echo "Downloading sample data"
curl -sS -o food_establishment_data.zip "https://docs.aws.amazon.com/emr/latest/
ManagementGuide/samples/food_establishment_data.zip" || handle_error "Failed to
    download sample data"

# Verify downloaded file
if [ ! -f food_establishment_data.zip ] || [ ! -s food_establishment_data.zip ];
    then
        handle_error "Downloaded file is empty or missing"
    fi

unzip -o food_establishment_data.zip || handle_error "Failed to unzip sample
    data"
echo "Sample data downloaded and extracted successfully."

# Secure the sample data file
chmod 600 food_establishment_data.csv

# Upload sample data to S3
echo "Uploading sample data to S3"
aws s3 cp food_establishment_data.csv "s3://$BUCKET_NAME/" --sse AES256 ||
    handle_error "Failed to upload sample data"
echo "Sample data uploaded successfully."

# Clean up sensitive local files
rm -f food_establishment_data.zip health_violations.py

# Create IAM default roles for EMR
echo "Creating IAM default roles for EMR"
aws emr create-default-roles 2>/dev/null || true
echo "IAM default roles created successfully."

# Check if EC2 key pair exists
echo "Checking for EC2 key pair"
KEY_PAIRS=$(aws ec2 describe-key-pairs --query "KeyPairs[*].KeyName" --output
    text 2>/dev/null || true)

if [ -z "$KEY_PAIRS" ]; then
```

```

    echo "No EC2 key pairs found. Creating a new key pair..."
    KEY_NAME="emr-tutorial-key-${RANDOM_ID}"
    KEY_NAME_FILE="${KEY_NAME}.pem"
    aws ec2 create-key-pair --key-name "$KEY_NAME" \
        --tag-specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=emr-gs}]' \
        --query "KeyMaterial" --output text > "$KEY_NAME_FILE"
    chmod 400 "$KEY_NAME_FILE"
    echo "Created new key pair: $KEY_NAME"
else
    # Use the first available key pair
    KEY_NAME=$(echo "$KEY_PAIRS" | awk '{print $1}')
    echo "Using existing key pair: $KEY_NAME"
fi

# Launch EMR cluster with security best practices
echo "Launching EMR cluster with Spark"
CLUSTER_RESPONSE=$(aws emr create-cluster \
    --name "EMR Tutorial Cluster" \
    --release-label emr-6.10.0 \
    --applications Name=Spark \
    --ec2-attributes KeyName="$KEY_NAME" \
    --instance-type m5.xlarge \
    --instance-count 3 \
    --use-default-roles \
    --log-uri "s3://$BUCKET_NAME/logs/" \
    --ebs-root-volume-size 100 \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=emr-gs \
    --security-configuration "EMR-Tutorial-SecurityConfig" 2>/dev/null || true)

# Check for errors in the response
if echo "$CLUSTER_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to create EMR cluster: $CLUSTER_RESPONSE"
fi

# Extract cluster ID using jq if available, otherwise use alternative parsing
if command -v jq &> /dev/null; then
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | jq -r '.ClusterId // empty')
else
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterId"[[:space:]]*:' \
        [[:space:]]*"^[^"]*" | grep -o 'j-[A-Z0-9]*' || true)
fi

if [ -z "$CLUSTER_ID" ] || [ "$CLUSTER_ID" == "null" ]; then

```

```
    handle_error "Failed to extract cluster ID from response: $CLUSTER_RESPONSE"
fi

echo "EMR cluster created with ID: $CLUSTER_ID"

# Wait for cluster to be ready
echo "Waiting for cluster to be ready (this may take several minutes)..."
aws emr wait cluster-running --cluster-id "$CLUSTER_ID" || handle_error "Cluster
failed to reach running state"

# Check if cluster is in WAITING state
CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query
"Cluster.Status.State" --output text)
if [ "$CLUSTER_STATE" != "WAITING" ]; then
    echo "Waiting for cluster to reach WAITING state..."
    WAIT_COUNT=0
    MAX_WAIT=120
    while [ "$CLUSTER_STATE" != "WAITING" ]; do
        if [ $WAIT_COUNT -ge $MAX_WAIT ]; then
            handle_error "Cluster did not reach WAITING state within timeout
period"
        fi
        sleep 30
        CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --
query "Cluster.Status.State" --output text)
        echo "Current cluster state: $CLUSTER_STATE"

        # Check for error states
        if [[ "$CLUSTER_STATE" == "TERMINATED_WITH_ERRORS" || "$CLUSTER_STATE" ==
"TERMINATED" ]]; then
            handle_error "Cluster entered error state: $CLUSTER_STATE"
        fi
        WAIT_COUNT=$((WAIT_COUNT + 1))
    done
fi

echo "Cluster is now in WAITING state and ready to accept work."

# Submit Spark application as a step
echo "Submitting Spark application as a step"
STEP_RESPONSE=$(aws emr add-steps \
    --cluster-id "$CLUSTER_ID" \
    --steps Type=Spark,Name="Health Violations
Analysis",ActionOnFailure=CONTINUE,Args=["s3://$BUCKET_NAME/
```

```
health_violations.py", "--data_source", "s3://$BUCKET_NAME/
food_establishment_data.csv", "--output_uri", "s3://$BUCKET_NAME/results/"]])

# Check for errors in the response
if echo "$STEP_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to submit step: $STEP_RESPONSE"
fi

# Extract step ID using appropriate method
if command -v jq &> /dev/null; then
    STEP_ID=$(echo "$STEP_RESPONSE" | jq -r '.StepIds[0] // empty')
else
    STEP_ID=$(echo "$STEP_RESPONSE" | grep -o 's-[A-Z0-9]*' | head -1 || true)
fi

if [ -z "$STEP_ID" ] || [ "$STEP_ID" == "null" ]; then
    echo "Full step response: $STEP_RESPONSE"
    handle_error "Failed to extract valid step ID from response"
fi

echo "Step submitted with ID: $STEP_ID"

# Wait for step to complete with timeout
echo "Waiting for step to complete (this may take several minutes)..."
aws emr wait step-complete --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID" ||
    handle_error "Step failed to complete"

# Check step status
STEP_STATE=$(aws emr describe-step --cluster-id "$CLUSTER_ID" --step-id
"$STEP_ID" --query "Step.Status.State" --output text)
if [ "$STEP_STATE" != "COMPLETED" ]; then
    handle_error "Step did not complete successfully. Final state: $STEP_STATE"
fi

echo "Step completed successfully."

# View results
echo "Listing output files in S3"
aws s3 ls "s3://$BUCKET_NAME/results/" || handle_error "Failed to list output
files"

# Download results
echo "Downloading results file"
```

```
RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-[0-9]*\.csv"
| head -1 || true)
if [ -z "$RESULT_FILE" ]; then
    echo "No result file found with pattern 'part-[0-9]*.csv'. Trying to find any
    CSV file..."
    RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-.*\.csv"
    | head -1 || true)
    if [ -z "$RESULT_FILE" ]; then
        echo "Listing all files in results directory:"
        aws s3 ls "s3://$BUCKET_NAME/results/"
        handle_error "No result file found in the output directory"
    fi
fi

aws s3 cp "s3://$BUCKET_NAME/results/$RESULT_FILE" ./results.csv --sse AES256 ||
    handle_error "Failed to download results file"
chmod 600 ./results.csv

echo "Results downloaded to results.csv"
echo "Top 10 establishments with the most red violations:"
cat results.csv

# Display SSH connection information
echo ""
echo "To connect to the cluster via SSH, use the following command:"
echo "aws emr ssh --cluster-id $CLUSTER_ID --key-pair-file ${KEY_NAME_FILE:-./
${KEY_NAME}.pem}"

# Display summary of created resources
echo ""
echo "======"
echo "RESOURCES CREATED"
echo "======"
echo "- S3 Bucket: $BUCKET_NAME"
echo "- EMR Cluster: $CLUSTER_ID"
echo "- Results file: results.csv"
if [ -f "${KEY_NAME_FILE:-}" ]; then
    echo "- EC2 Key Pair: $KEY_NAME (saved to ${KEY_NAME_FILE})"
fi

# Perform cleanup
cleanup
```

```
echo "Script completed successfully."
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dei comandi della AWS CLI .
 - [AddSteps](#)
 - [Cap](#)
 - [CreateCluster](#)
 - [CreateDefaultRoles](#)
 - [CreateKeyPair](#)
 - [DescribeCluster](#)
 - [DescribeKeyPairs](#)
 - [DescribeStep](#)
 - [Ls](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)
 - [Ssh](#)
 - [TerminateClusters](#)
 - [Attendi](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Guida introduttiva alla gestione della configurazione

L'esempio di codice seguente mostra come:

- Crea un bucket Amazon S3
- Creare un argomento Amazon SNS.
- Creare un ruolo IAM per Config
- Configurare il registratore di configurazione Config

- Avvia il registratore di configurazione
- Verifica la configurazione di Config

Bash

AWS CLI con lo script Bash

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel [repository dei tutorial sugli esempi di codice per gli sviluppatori](#).

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}
```

```
# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
"true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
"$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
"true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
"$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi
    fi
}
```



```

        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "=====
echo "CREATED RESOURCES"
    echo "=====
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "=====
}

```

```
# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
    text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1" # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
    fi
fi
```

```

fi
check_command "$BUCKET_RESULT"
echo "S3 bucket created: $S3_BUCKET_NAME"

aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
echo "Tags applied to S3 bucket"
else
echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
"BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true"
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-`${RANDOM_ID}`"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
    Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-`${RANDOM_ID}`"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "config.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}

```

```
]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam:[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    }
  ]
}
```

```

    },
    {
        "Effect": "Allow",
        "Action": [
            "sns:Publish"
        ],
        "Resource": "${SNS_TOPIC_ARN}"
    }
]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
"$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-records 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
| head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
    "name": "${CONFIG_RECORDER_NAME}",

```

```

    "roleARN": "${ROLE_ARN}",
    "recordingMode": {
        "recordingFrequency": "CONTINUOUS"
    }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
    "allSupported": true,
    "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
    Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
| head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel

```

```
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi
```

```
# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "=====
echo "CLEANUP CONFIRMATION"
echo "=====
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
```



```
cleanup_resources
echo "Cleanup completed."
else
    echo "Resources will not be cleaned up. You can manually clean them up
    later."
fi

echo "Script completed successfully!"
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dei comandi della AWS CLI .

- [AttachRolePolicy](#)
- [CreateBucket](#)
- [CreateRole](#)
- [CreateTopic](#)
- [DeleteBucket](#)
- [DeleteConfigurationRecorder](#)
- [DeleteDeliveryChannel](#)
- [DeleteRole](#)
- [DeleteRolePolicy](#)
- [DeleteTopic](#)
- [DescribeConfigurationRecorderStatus](#)
- [DescribeConfigurationRecorders](#)
- [DescribeDeliveryChannels](#)
- [DetachRolePolicy](#)
- [GetCallerIdentity](#)
- [PutConfigurationRecorder](#)
- [PutDeliveryChannel](#)
- [PutPublicAccessBlock](#)
- [PutRolePolicy](#)
- [Rm](#)
- [StartConfigurationRecorder](#)
- [StopConfigurationRecorder](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Guida introduttiva all'estrazione del testo dei documenti

L'esempio di codice seguente mostra come:

- Creare un bucket S3
- Carica un documento su S3
- Eseguire la pulizia delle risorse

Bash

AWS CLI con lo script Bash

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel [repository dei tutorial sugli esempi di codice per gli sviluppatori](#).

```
#!/bin/bash

# Amazon Textract Getting Started Tutorial Script
# This script demonstrates how to use Amazon Textract to analyze document text

set -euo pipefail

# Set up logging with restricted permissions
LOG_FILE="textract-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "=====
echo "Amazon Textract Getting Started Tutorial"
echo "=====
```

```
echo "This script will guide you through using Amazon Textract to analyze
document text."
echo ""

# Function to check for errors in command output and exit code
check_error() {
    local exit_code=$1
    local output=$2
    local cmd=$3

    if [ $exit_code -ne 0 ] || echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output" | sed 's/\(aws_secret_access_key\|Authorization\|X-Amz-
Security-Token\).*\/1=***REDACTED***\/g'
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    # Clean up temporary JSON files
    if [ -f "document.json" ]; then
        rm -f document.json
    fi

    if [ -f "features.json" ]; then
        rm -f features.json
    fi

    if [ -n "${DOCUMENT_NAME:-}" ] && [ -n "${BUCKET_NAME:-}" ]; then
        echo "Deleting document from S3..."
        aws s3 rm "s3://${BUCKET_NAME}/${DOCUMENT_NAME}" || echo "Failed to
delete document"
    fi

    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" = "false" ];
then
        echo "Deleting S3 bucket..."
        aws s3 rb "s3://${BUCKET_NAME}" --force || echo "Failed to delete bucket"
    fi
}
```

```
# Set up trap for cleanup on exit
trap cleanup_on_error EXIT

# Verify AWS CLI is installed and configured
echo "Verifying AWS CLI configuration..."
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed."
    exit 1
fi

AWS_CONFIG_OUTPUT=$(aws configure list 2>&1)
AWS_CONFIG_STATUS=$?
if [ $AWS_CONFIG_STATUS -ne 0 ]; then
    echo "ERROR: AWS CLI is not properly configured."
    echo "$AWS_CONFIG_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization
\).*\/1=***REDACTED***\/g'
    exit 1
fi

# Verify AWS region is configured and supports Textract
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Please run 'aws configure' to set a
default region."
    exit 1
fi

# Check if Textract is available in the configured region
echo "Checking if Amazon Textract is available in region $AWS_REGION..."
TEXTRACT_CHECK=$(aws textract help 2>&1)
TEXTRACT_CHECK_STATUS=$?
if [ $TEXTRACT_CHECK_STATUS -ne 0 ]; then
    echo "ERROR: Amazon Textract may not be available in region $AWS_REGION."
    exit 1
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null || echo "")
```

```
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="textract-`${RANDOM_ID}`"
fi
DOCUMENT_NAME="document.png"
RESOURCES_CREATED=()

# Step 1: Create S3 bucket
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
    CREATE_BUCKET_OUTPUT=$(aws s3 mb "s3://$BUCKET_NAME" --region "$AWS_REGION"
2>&1)
    CREATE_BUCKET_STATUS=$?
    echo "$CREATE_BUCKET_OUTPUT"
    check_error $CREATE_BUCKET_STATUS "$CREATE_BUCKET_OUTPUT" "aws s3 mb s3://
$BUCKET_NAME"

    aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-textract-gs}]'

    # Apply security settings to bucket
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled 2>&1 || true
    aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" --server-side-
encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault":
{"SSEAlgorithm": "AES256"}}]}' 2>&1 || true
    aws s3api put-bucket-acl --bucket "$BUCKET_NAME" --acl private 2>&1 || true

    RESOURCES_CREATED+=("S3 Bucket: $BUCKET_NAME")
fi

# Step 2: Check if sample document exists, if not create a simple one
if [ ! -f "$DOCUMENT_NAME" ]; then
    echo "Sample document not found. Generating a sample document..."

    # Create a simple PNG document using ImageMagick or convert
    if command -v convert &> /dev/null; then
```

```

        convert -size 400x300 xc:white -pointsize 20 -fill black -draw "text
50,50 'Sample Document'" "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Generated sample document: $DOCUMENT_NAME"
    else
        # Fallback: create a minimal valid PNG using base64
        echo "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAyAAAFfcSjAAAAADULEQVR42mNk
+M9QDwAdhgGAWjR9awAAAABJRU5ErkJggg==" | base64 -d > "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Created minimal sample document: $DOCUMENT_NAME"
    fi
fi

# Step 3: Upload document to S3
echo "Uploading document to S3..."
UPLOAD_OUTPUT=$(aws s3 cp "$DOCUMENT_NAME" "s3://$BUCKET_NAME/" --sse AES256
2>&1)
UPLOAD_STATUS=$?
echo "$UPLOAD_OUTPUT"
check_error $UPLOAD_STATUS "$UPLOAD_OUTPUT" "aws s3 cp "$DOCUMENT_NAME" s3://
$BUCKET_NAME/"
RESOURCES_CREATED+=("S3 Object: s3://$BUCKET_NAME/$DOCUMENT_NAME")

# Step 4: Analyze document with Amazon Textract
echo "Analyzing document with Amazon Textract..."
echo "This may take a few seconds..."

# Create a JSON file for the document parameter to avoid shell escaping issues
cat > document.json << 'EOF'
{
    "S3Object": {
        "Bucket": "BUCKET_PLACEHOLDER",
        "Name": "DOCUMENT_PLACEHOLDER"
    }
}
EOF

sed -i.bak "s|BUCKET_PLACEHOLDER|$BUCKET_NAME|g; s|DOCUMENT_PLACEHOLDER|
$DOCUMENT_NAME|g" document.json
rm -f document.json.bak
chmod 600 document.json

# Create a JSON file for the feature types parameter
cat > features.json << 'EOF'

```

```

["TABLES", "FORMS", "SIGNATURES"]
EOF
chmod 600 features.json

ANALYZE_OUTPUT=$(aws textract analyze-document --document file://document.json --
feature-types file://features.json 2>&1)
ANALYZE_STATUS=$?

echo "Analysis complete."
if [ $ANALYZE_STATUS -ne 0 ]; then
    echo "ERROR: Document analysis failed"
    echo "$ANALYZE_OUTPUT" | sed 's/(aws_secret_access_key\|Authorization\|Token
\).*/\1=***REDACTED***/g'
    exit 1
fi

# Save the analysis results to a file with restricted permissions
echo "$ANALYZE_OUTPUT" > textract-analysis-results.json
chmod 600 textract-analysis-results.json
echo "Analysis results saved to textract-analysis-results.json"
RESOURCES_CREATED+=("Local file: textract-analysis-results.json")

# Display a summary of the analysis
echo ""
echo "=====
echo "Analysis Summary"
echo "=====
PAGES=$(echo "$ANALYZE_OUTPUT" | grep -o '"Pages": [0-9]*' | head -1 | awk
'{print $2}')
echo "Document pages: $PAGES"

BLOCKS_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType":' | wc -l)
echo "Total blocks detected: $BLOCKS_COUNT"

# Count different block types using jq if available, fallback to grep
PAGE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "PAGE"' | wc -l ||
echo 0)
LINE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "LINE"' | wc -l ||
echo 0)
WORD_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "WORD"' | wc -l ||
echo 0)
TABLE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "TABLE"' | wc -l ||
echo 0)

```

```

CELL_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "CELL"' | wc -l ||
echo 0)
KEY_VALUE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "KEY_VALUE_SET"'
| wc -l || echo 0)
SIGNATURE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "SIGNATURE"' |
wc -l || echo 0)

echo "Pages: $PAGE_COUNT"
echo "Lines of text: $LINE_COUNT"
echo "Words: $WORD_COUNT"
echo "Tables: $TABLE_COUNT"
echo "Table cells: $CELL_COUNT"
echo "Key-value pairs: $KEY_VALUE_COUNT"
echo "Signatures: $SIGNATURE_COUNT"
echo ""

# Cleanup confirmation
echo ""
echo "=====
echo "RESOURCES CREATED"
echo "=====
for resource in "${RESOURCES_CREATED[@]}"; do
    echo "- $resource"
done
echo ""
echo "=====
echo "CLEANUP CONFIRMATION"
echo "=====
echo "Cleaning up resources..."

# Delete document from S3
echo "Deleting document from S3..."
DELETE_DOC_OUTPUT=$(aws s3 rm "s3://$BUCKET_NAME/$DOCUMENT_NAME" 2>&1)
DELETE_DOC_STATUS=$?
echo "$DELETE_DOC_OUTPUT"
check_error $DELETE_DOC_STATUS "$DELETE_DOC_OUTPUT" "aws s3 rm s3://$BUCKET_NAME/
$DOCUMENT_NAME"

# Delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Deleting S3 bucket..."
    DELETE_BUCKET_OUTPUT=$(aws s3 rb "s3://$BUCKET_NAME" --force 2>&1)
    DELETE_BUCKET_STATUS=$?
    echo "$DELETE_BUCKET_OUTPUT"

```



```
    check_error $DELETE_BUCKET_STATUS "$DELETE_BUCKET_OUTPUT" "aws s3 rb s3://
$BUCKET_NAME --force"
fi

# Delete local JSON files
rm -f document.json features.json

echo "Cleanup complete. The analysis results file (textract-analysis-
results.json) has been kept."

echo ""
echo "=====
echo "Tutorial complete!"
echo "=====
echo "You have successfully analyzed a document using Amazon Textract."
echo "The analysis results are available in textract-analysis-results.json"
echo ""

trap - EXIT
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dei comandi della AWS CLI .
 - [AnalyzeDocument](#)
 - [Cap](#)
 - [Aiuto](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Guida introduttiva agli store di funzionalità di machine learning

L'esempio di codice seguente mostra come:

- Impostare autorizzazioni IAM
- Crea un ruolo di SageMaker esecuzione

- Creazione di gruppi di funzionalità
- Eseguire la pulizia delle risorse

Bash

AWS CLI con lo script Bash

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel [repository dei tutorial sugli esempi di codice per gli sviluppatori](#).

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
```

```

        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
"$resource_name"
                ;;

```

```

        "S3Bucket")
        echo "Emptying S3 bucket: $resource_name"
        aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
        echo "Deleting S3 bucket: $resource_name"
        aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
        ;;
    "IAMRole")
        echo "Detaching policies from role: $resource_name"
        aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
        aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
        echo "Deleting IAM role: $resource_name"
        aws iam delete-role --role-name "$resource_name" 2>/dev/null
        ;;
    *)
        echo "Unknown resource type: $resource_type"
        ;;
esac
done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \

```

```
--role-name "$role_name" \  
--assume-role-policy-document "$trust_policy" \  
--description "SageMaker execution role for Feature Store tutorial" 2>&1)  
  
if echo "$role_result" | grep -i "error" > /dev/null; then  
    handle_error "Failed to create IAM role: $role_result"  
fi  
  
echo "Role created successfully" >&2  
CREATED_RESOURCES+=("IAMRole:$role_name")  
  
# Tag the role  
echo "Tagging IAM role..." >&2  
aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith  
Key=tutorial,Value=sagemaker-featurestore 2>&1  
  
# Attach necessary policies  
echo "Attaching policies to role..." >&2  
  
# SageMaker execution policy  
local policy1_result=$(aws iam attach-role-policy \  
    --role-name "$role_name" \  
    --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)  
  
if echo "$policy1_result" | grep -i "error" > /dev/null; then  
    handle_error "Failed to attach SageMaker policy: $policy1_result"  
fi  
  
# S3 access policy  
local policy2_result=$(aws iam attach-role-policy \  
    --role-name "$role_name" \  
    --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)  
  
if echo "$policy2_result" | grep -i "error" > /dev/null; then  
    handle_error "Failed to attach S3 policy: $policy2_result"  
fi  
  
# Get account ID for role ARN  
local account_id=$(aws sts get-caller-identity --query Account --output text)  
local role_arn="arn:aws:iam::${account_id}:role/${role_name}"  
  
echo "Role ARN: $role_arn" >&2  
echo "Waiting 10 seconds for role to propagate..." >&2  
sleep 10
```

```
# Return only the role ARN to stdout
echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"
```

```
# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
            --region "$REGION" 2>&1)
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
    fi

    if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
        echo "Failed to create S3 bucket: $BUCKET_RESULT"
        exit 1
    fi

    echo "$BUCKET_RESULT"
    CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")
```

```
# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
  'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
  --bucket "$S3_BUCKET_NAME" \
  --public-access-block-configuration
  "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
  echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
  cleanup_resources
  exit 1
fi
else
  echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
  --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
  --record-identifier-feature-name "customer_id" \
  --event-time-feature-name "EventTime" \
  --feature-definitions '[
    {"FeatureName": "customer_id", "FeatureType": "Integral"},
    {"FeatureName": "name", "FeatureType": "String"},
    {"FeatureName": "age", "FeatureType": "Integral"},
    {"FeatureName": "address", "FeatureType": "String"},
    {"FeatureName": "membership_type", "FeatureType": "String"},
    {"FeatureName": "EventTime", "FeatureType": "Fractional"}
  ]' \
  --online-store-config '{"EnableOnlineStore": true}' \
  --offline-store-config '{
    "S3StorageConfig": {
```



```

        "S3Uri": "s3://${S3_BUCKET_NAME}/${PREFIX}"
    },
    "DisableGlueTableCreation": false
}' \
--role-arn "$ROLE_ARN" \
--tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://${S3_BUCKET_NAME}/${PREFIX}"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
2>&1)

```

```
if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
```

```
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
```

```

        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS"
| grep -i "Records" > /dev/null; then

```

```

        echo "Failed to get batch records: $BATCH_RECORDS"
        cleanup_resources
        exit 1
    fi

    echo "$BATCH_RECORDS"

    # List feature groups
    echo "Listing feature groups:"
    FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

    if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
        echo "Failed to list feature groups: $FEATURE_GROUPS"
        cleanup_resources
        exit 1
    fi

    echo "$FEATURE_GROUPS"

    # Display summary of created resources
    echo ""
    echo "======"
    echo "TUTORIAL COMPLETED SUCCESSFULLY!"
    echo "======"
    echo "Resources created:"
    echo "- S3 Bucket: $S3_BUCKET_NAME"
    echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
    echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
    if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
        echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
    fi
    echo ""
    echo "You can now:"
    echo "1. View your feature groups in the SageMaker console"
    echo "2. Query the offline store using Amazon Athena"
    echo "3. Use the feature groups in your ML workflows"
    echo "======"
    echo ""

    # Handle cleanup
    if [ "$AUTO_CLEANUP" = "y" ]; then
        echo "Auto-cleanup enabled. Starting cleanup..."
        cleanup_resources
    fi

```

```

    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "======"
    echo "CLEANUP CONFIRMATION"
    echo "======"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
$ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o
'IAMRole:[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
fi

echo "Script completed at $(date)"

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dei comandi della AWS CLI .
 - [AttachRolePolicy](#)
 - [CreateBucket](#)
 - [CreateFeatureGroup](#)
 - [CreateRole](#)
 - [DeleteBucket](#)
 - [DeleteFeatureGroup](#)

- [DeleteRole](#)
- [DescribeFeatureGroup](#)
- [DetachRolePolicy](#)
- [GetCallerIdentity](#)
- [GetRole](#)
- [ListFeatureGroups](#)
- [PutPublicAccessBlock](#)
- [Rm](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Guida introduttiva allo storage di oggetti

L'esempio di codice seguente mostra come:

- Crea il tuo primo bucket S3
- Caricamento di un oggetto
- Abilita il controllo delle versioni
- Configura la crittografia predefinita
- Aggiungi tag al tuo bucket
- Elenca oggetti e versioni
- Eseguire la pulizia delle risorse

Bash

AWS CLI con lo script Bash

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel [repository dei tutorial sugli esempi di codice per gli sviluppatori](#).

```
#!/bin/bash
# S3 Getting Started - Create a bucket, upload and download objects, copy to a
# folder prefix, enable versioning, configure encryption and public access
# blocking, tag the bucket, list objects and versions, and clean up.

set -eE
set -o pipefail

# =====
# Prerequisites check
# =====

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
"$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Run 'aws configure' or set
AWS_DEFAULT_REGION."
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid. Run 'aws configure'."
    exit 1
fi

# =====
# Setup: logging, temp directory, resource tracking
# =====

UNIQUE_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null || true)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="s3api-${UNIQUE_ID}"

```



```

fi

TEMP_DIR=$(mktemp -d)
trap 'rm -rf "$TEMP_DIR"' EXIT
LOG_FILE="${TEMP_DIR}/s3-gettingstarted.log"
CREATED_RESOURCES=()

exec > >(tee -a "$LOG_FILE") 2>&1

echo "====="
echo "S3 Getting Started"
echo "====="
echo "Bucket name: ${BUCKET_NAME}"
echo "Temp directory: ${TEMP_DIR}"
echo "Log file: ${LOG_FILE}"
echo ""

# =====
# Helper functions
# =====

get_region() {
    echo "${AWS_REGION:-${AWS_DEFAULT_REGION:-${CONFIGURED_REGION}}}"
}

delete_object_versions() {
    local bucket=$1
    local query=$2

    local versions
    versions=$(aws s3api list-object-versions \
        --bucket "$bucket" \
        --query "$query" \
        --output json 2>&1) || return 0

    if [ -z "$versions" ] || [ "$versions" = "null" ] || [ "$versions" = "[]" ];
    then
        return 0
    fi

    echo "$versions" | jq -r '.[ ] | "\(.Key)\t\(.VersionId)"' 2>/dev/null | while
IFS=$'\t' read -r key version_id; do
    if [ -n "$key" ] && [ "$key" != "null" ]; then

```

```

        aws s3api delete-object --bucket "$bucket" --key "$key" --version-id
"$version_id" >/dev/null 2>&1 || true
    fi
done

return 0
}

# =====
# Error handling and cleanup functions
# =====

cleanup() {
    echo ""
    echo "=====
echo "CLEANUP"
    echo "=====

    if [ "$BUCKET_IS_SHARED" = "false" ]; then
        echo "Deleting all object versions in bucket..."

        delete_object_versions "$BUCKET_NAME" "Versions[.
{Key:Key,VersionId:VersionId}" || true

        delete_object_versions "$BUCKET_NAME" "DeleteMarkers[.
{Key:Key,VersionId:VersionId}" || true

        echo "Deleting bucket: ${BUCKET_NAME}"
        if ! aws s3api delete-bucket --bucket "$BUCKET_NAME" 2>/dev/null; then
            echo "WARNING: Failed to delete bucket ${BUCKET_NAME}"
        fi

        # Clean up logs bucket
        LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
        if aws s3api head-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/null; then
            echo "Deleting log bucket: ${LOG_TARGET_BUCKET}"
            if ! aws s3api delete-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/
null; then
                echo "WARNING: Failed to delete bucket ${LOG_TARGET_BUCKET}"
            fi
        fi
    else
        echo "Keeping shared bucket: ${BUCKET_NAME}"
    fi
}

```

```

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    local line_number=$1
    echo ""
    echo "=====
    echo "ERROR on line ${line_number}"
    echo "=====
    echo ""
    echo "Resources created before error:"
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        for RESOURCE in "${CREATED_RESOURCES[@}"; do
            echo "  - ${RESOURCE}"
        done
    else
        echo "  (none)"
    fi
    echo ""
    echo "Attempting cleanup..."
    cleanup
    exit 1
}

trap 'handle_error "$LINENO"' ERR

# =====
# Step 1: Create a bucket
# =====

echo "Step 1: Creating bucket ${BUCKET_NAME}..."
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
        if ! aws s3api create-bucket --bucket "$BUCKET_NAME" >/dev/null 2>&1;
    then
        echo "ERROR: Failed to create bucket $BUCKET_NAME"
        exit 1
    fi
else
    if ! aws s3api create-bucket \
        --bucket "$BUCKET_NAME" \

```

```

        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
2>&1; then
    echo "ERROR: Failed to create bucket $BUCKET_NAME in region $REGION"
    exit 1
fi
fi
CREATED_RESOURCES+=("s3:bucket:${BUCKET_NAME}")
echo "Bucket created."

if ! aws s3api put-bucket-tagging \
    --bucket "$BUCKET_NAME" \
    --tagging '{
        "TagSet": [
            {
                "Key": "project",
                "Value": "doc-smith"
            },
            {
                "Key": "tutorial",
                "Value": "s3-gettingstarted"
            }
        ]
    }' >/dev/null 2>&1; then
    echo "WARNING: Failed to tag bucket"
fi
fi
echo ""

# =====
# Step 2: Upload a sample text file
# =====

echo "Step 2: Uploading a sample text file..."

SAMPLE_FILE="${TEMP_DIR}/sample.txt"
cat > "$SAMPLE_FILE" << 'EOF'
Hello, Amazon S3! This is a sample file for the getting started tutorial.
EOF

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \

```

```
--server-side-encryption AES256 \  
--metadata "tutorial=s3-gettingstarted" >/dev/null 2>&1; then  
echo "ERROR: Failed to upload sample.txt"  
exit 1  
fi  
echo "File uploaded."  
echo ""  
  
# =====  
# Step 3: Download the object  
# =====  
  
echo "Step 3: Downloading the object..."  
  
DOWNLOAD_FILE="${TEMP_DIR}/downloaded-sample.txt"  
if ! aws s3api get-object \  
    --bucket "$BUCKET_NAME" \  
    --key "sample.txt" \  
    "$DOWNLOAD_FILE" >/dev/null 2>&1; then  
    echo "ERROR: Failed to download sample.txt"  
    exit 1  
fi  
echo "Downloaded to: ${DOWNLOAD_FILE}"  
echo "Contents:"  
cat "$DOWNLOAD_FILE"  
echo ""  
  
# =====  
# Step 4: Copy the object to a folder prefix  
# =====  
  
echo "Step 4: Copying object to a folder prefix..."  
  
if ! aws s3api copy-object \  
    --bucket "$BUCKET_NAME" \  
    --copy-source "${BUCKET_NAME}/sample.txt" \  
    --key "backup/sample.txt" \  
    --server-side-encryption AES256 \  
    --metadata-directive COPY >/dev/null 2>&1; then  
    echo "ERROR: Failed to copy object to backup/sample.txt"  
    exit 1  
fi  
echo "Object copied to backup/sample.txt."  
echo ""
```

```
# =====
# Step 5: Enable versioning and upload a second version
# =====

echo "Step 5: Enabling versioning..."

if ! aws s3api put-bucket-versioning \
    --bucket "$BUCKET_NAME" \
    --versioning-configuration Status=Enabled >/dev/null 2>&1; then
    echo "ERROR: Failed to enable versioning"
    exit 1
fi
echo "Versioning enabled."

echo "Uploading a second version of sample.txt..."
cat > "$SAMPLE_FILE" << 'EOF'
Hello, Amazon S3! This is version 2 of the sample file.
EOF

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted,version=2" >/dev/null 2>&1; then
    echo "ERROR: Failed to upload second version of sample.txt"
    exit 1
fi
echo "Second version uploaded."
echo ""

# =====
# Step 6: Configure SSE-S3 encryption
# =====

echo "Step 6: Configuring SSE-S3 default encryption..."

if ! aws s3api put-bucket-encryption \
    --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
```

```

        "SSEAlgorithm": "AES256"
    },
    "BucketKeyEnabled": true
}
]
}' >/dev/null 2>&1; then
echo "ERROR: Failed to configure SSE-S3 encryption"
exit 1
fi
echo "SSE-S3 encryption configured."
echo ""

# =====
# Step 7: Block all public access
# =====

echo "Step 7: Blocking all public access..."

if ! aws s3api put-public-access-block \
    --bucket "$BUCKET_NAME" \
    --public-access-block-configuration '{
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to block public access"
    exit 1
fi
echo "Public access blocked."
echo ""

# =====
# Step 8: Configure bucket logging
# =====

echo "Step 8: Configuring bucket logging..."

LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
        aws s3api create-bucket --bucket "$LOG_TARGET_BUCKET" >/dev/null 2>&1 ||
true

```

```

else
    aws s3api create-bucket \
        --bucket "$LOG_TARGET_BUCKET" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
2>&1 || true
fi

if ! aws s3api put-bucket-tagging \
    --bucket "$LOG_TARGET_BUCKET" \
    --tagging '{
        "TagSet": [
            {
                "Key": "project",
                "Value": "doc-smith"
            },
            {
                "Key": "tutorial",
                "Value": "s3-gettingstarted"
            }
        ]
    }' >/dev/null 2>&1; then
    echo "WARNING: Failed to tag log bucket"
fi

aws s3api put-bucket-acl --bucket "$LOG_TARGET_BUCKET" --acl log-delivery-
write 2>/dev/null || true

if ! aws s3api put-bucket-logging \
    --bucket "$BUCKET_NAME" \
    --bucket-logging-status '{
        "LoggingEnabled": {
            "TargetBucket": "'$LOG_TARGET_BUCKET'",
            "TargetPrefix": "logs/"
        }
    }' >/dev/null 2>&1; then
    echo "WARNING: Failed to configure bucket logging"
else
    echo "Bucket logging configured."
fi
else
    echo "Skipping logging configuration for shared bucket."
fi
echo ""

```



```
# =====
# Step 9: Tag the bucket
# =====

echo "Step 9: Tagging the bucket..."

if ! aws s3api put-bucket-tagging \
    --bucket "$BUCKET_NAME" \
    --tagging '{
        "TagSet": [
            {
                "Key": "project",
                "Value": "doc-smith"
            },
            {
                "Key": "tutorial",
                "Value": "s3-gettingstarted"
            },
            {
                "Key": "Environment",
                "Value": "Tutorial"
            },
            {
                "Key": "Project",
                "Value": "S3-GettingStarted"
            },
            {
                "Key": "ManagedBy",
                "Value": "Bash-Tutorial"
            }
        ]
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to tag bucket"
    exit 1
fi
echo "Bucket tagged."

echo "Verifying tags..."
if ! aws s3api get-bucket-tagging --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to retrieve bucket tags"
fi
echo ""
```

```

# =====
# Step 10: List objects and versions
# =====

echo "Step 10: Listing objects..."

if ! aws s3api list-objects-v2 --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list objects"
fi
echo ""

echo "Listing object versions..."

if ! aws s3api list-object-versions --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list object versions"
fi
echo ""

# =====
# Step 11: Cleanup
# =====

echo ""
echo "=====
echo "TUTORIAL COMPLETE"
echo "=====
echo ""
echo "Resources created:"
if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
    for RESOURCE in "${CREATED_RESOURCES[@]}"; do
        echo "  - ${RESOURCE}"
    done
else
    echo "  (none)"
fi
echo ""
echo "=====
echo "CLEANUP"
echo "=====
echo "Cleaning up all created resources..."
cleanup

echo ""

```

```
echo "Done."
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dei comandi della AWS CLI .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [HeadObject](#)
 - [ListObjectVersions](#)
 - [ListObjectsV2](#)
 - [PutBucketEncryption](#)
 - [PutBucketTagging](#)
 - [PutBucketVersioning](#)
 - [PutObject](#)
 - [PutPublicAccessBlock](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Guida introduttiva all'analisi delle query

L'esempio di codice seguente mostra come:

- Crea un bucket S3 per i risultati delle query
- Creare un database
- Creare una tabella
- Esecuzione di una query
- Crea e usa interrogazioni denominate
- Eseguire la pulizia delle risorse

Bash

AWS CLI con lo script Bash

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel [repository dei tutorial sugli esempi di codice per gli sviluppatori](#).

```
#!/bin/bash

# Amazon Athena Getting Started Script
# This script demonstrates how to use Amazon Athena with AWS CLI
# It creates a database, table, runs queries, and manages named queries

set -euo pipefail

# Security: Validate AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
    echo "ERROR: AWS credentials not configured or invalid"
    exit 1
fi

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="athena-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Athena Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${NAMED_QUERY_ID:-}" ]; then
```

```

        echo "- Named Query: $NAMED_QUERY_ID"
    fi
    if [ -n "${DATABASE_NAME:-}" ]; then
        echo "- Database: $DATABASE_NAME"
        if [ -n "${TABLE_NAME:-}" ]; then
            echo "- Table: $TABLE_NAME in $DATABASE_NAME"
        fi
    fi
    if [ -n "${S3_BUCKET:-}" ]; then
        echo "- S3 Bucket: $S3_BUCKET"
    fi

    echo "Exiting..."
    exit 1
}

# Security: Validate bucket name format
validate_bucket_name() {
    local bucket_name="$1"
    if [[ ! "$bucket_name" =~ ^[a-z0-9][a-z0-9.-]*[a-z0-9]$ ]] ||
    [ ${#bucket_name} -lt 3 ] || [ ${#bucket_name} -gt 63 ]; then
        return 1
    fi
    return 0
}

# Security: Validate database and table names
validate_identifier() {
    local identifier="$1"
    if [[ ! "$identifier" =~ ^[a-zA-Z][a-zA-Z0-9_]*$ ]]; then
        return 1
    fi
    return 0
}

# Security: Safely generate random identifier
if ! command -v openssl &>/dev/null; then
    RANDOM_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
else
    RANDOM_ID=$(openssl rand -hex 6)
fi

# Security: Validate random ID format
if [[ ! "$RANDOM_ID" =~ ^[a-f0-9]{12}$ ]]; then

```

```
    handle_error "Failed to generate valid random ID"
fi

# Check for shared prereq bucket with proper error handling
PREREQ_BUCKET=""
if aws cloudformation describe-stacks --stack-name tutorial-prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
    text 2>/dev/null | grep -qv "^$"; then
    PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-
prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --
output text 2>/dev/null)
fi

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET="athena-${RANDOM_ID}"
fi

if ! validate_bucket_name "$S3_BUCKET"; then
    handle_error "Invalid S3 bucket name: $S3_BUCKET"
fi

DATABASE_NAME="mydatabase"
TABLE_NAME="cloudfront_logs"

if ! validate_identifier "$DATABASE_NAME"; then
    handle_error "Invalid database name: $DATABASE_NAME"
fi

if ! validate_identifier "$TABLE_NAME"; then
    handle_error "Invalid table name: $TABLE_NAME"
fi

# Get the current AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No AWS region found in configuration, defaulting to $AWS_REGION"
fi
```

```
# Security: Validate region format - expanded regex for newer regions
if [[ ! "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]] && [[ ! "$AWS_REGION" =~
  ^[a-z]+-[a-z]+-[0-9]{1}$ ]]; then
    echo "WARNING: Region format may be invalid: $AWS_REGION"
fi

echo "Using AWS Region: $AWS_REGION"

# Create S3 bucket for Athena query results
echo "Creating S3 bucket for Athena query results: $S3_BUCKET"
if [ "$BUCKET_IS_SHARED" = false ]; then
    CREATE_BUCKET_RESULT=$(aws s3 mb "s3://$S3_BUCKET" --region "$AWS_REGION"
2>&1)
    if echo "$CREATE_BUCKET_RESULT" | grep -qi "error\|failed"; then
        handle_error "Failed to create S3 bucket: $CREATE_BUCKET_RESULT"
    fi

    aws s3api put-bucket-tagging \
        --bucket "$S3_BUCKET" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-athena-gs}]'

# Security: Enable S3 bucket encryption with KMS validation
echo "Enabling default encryption on S3 bucket..."
if ! aws s3api put-bucket-encryption \
    --bucket "$S3_BUCKET" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' 2>&1; then
    echo "Warning: Could not enable encryption on bucket"
fi

# Security: Block public access
echo "Blocking public access to S3 bucket..."
if ! aws s3api put-public-access-block \
    --bucket "$S3_BUCKET" \
    --public-access-block-configuration \
```

```

"BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
2>&1; then
    echo "Warning: Could not block public access on bucket"
fi

# Security: Enable versioning for data protection
echo "Enabling versioning on S3 bucket..."
if ! aws s3api put-bucket-versioning \
    --bucket "$S3_BUCKET" \
    --versioning-configuration Status=Enabled 2>&1; then
    echo "Warning: Could not enable versioning on bucket"
fi

echo "S3 bucket created successfully: $S3_BUCKET"
fi

# Step 1: Create a database
echo "Step 1: Creating Athena database: $DATABASE_NAME"
CREATE_DB_RESULT=$(aws athena start-query-execution \
    --query-string "CREATE DATABASE IF NOT EXISTS $DATABASE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create database: $CREATE_DB_RESULT"
fi

QUERY_ID=$(echo "$CREATE_DB_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/
null || echo "$CREATE_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*" | cut -
d'"'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from database creation response"
fi
echo "Database creation query ID: $QUERY_ID"

# Wait for database creation to complete
echo "Waiting for database creation to complete..."
WAIT_TIMEOUT=60
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \

```



```

        --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Database creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Database creation failed with status: $QUERY_STATUS"
    fi
    echo "Database creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Database creation timed out"
fi

# Verify the database was created
echo "Verifying database creation..."
LIST_DB_RESULT=$(aws athena list-databases --catalog-name AwsDataCatalog --region
"$AWS_REGION" 2>&1)
if echo "$LIST_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list databases: $LIST_DB_RESULT"
fi
echo "$LIST_DB_RESULT"

# Step 2: Create a table
echo "Step 2: Creating Athena table: $TABLE_NAME"
# Replace the region placeholder in the S3 location
CREATE_TABLE_QUERY="CREATE EXTERNAL TABLE IF NOT EXISTS $DATABASE_NAME.
$TABLE_NAME (
    \`Date\` DATE,
    Time STRING,
    Location STRING,
    Bytes INT,
    RequestIP STRING,
    Method STRING,
    Host STRING,
    Uri STRING,
    Status INT,
    Referrer STRING,
    os STRING,
    Browser STRING,

```

```

    BrowserVersion STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
    \"input.regex\" = \"^(?!#)([ ]+)\s+([ ]+)\s+([ ]+)\s+([ ]+)\s+
+([ ]+)\s+([ ]+)\s+([ ]+)\s+([ ]+)\s+([ ]+)\s+([ ]+)\s+([ ]+)\s+
\\\\([ ]+\\\\\\\\([ ]+\\\\\\\\;)+).*\\\\\\\\%20([ ]+\\\\\\\\/)+[\\\\\\\\/](.*)$\"
) LOCATION 's3://athena-examples-us-east-1/cloudfront/plaintext/';"

CREATE_TABLE_RESULT=$(aws athena start-query-execution \
    --query-string "$CREATE_TABLE_QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_TABLE_RESULT" | grep -qi "error\\|failed"; then
    handle_error "Failed to create table: $CREATE_TABLE_RESULT"
fi

QUERY_ID=$(echo "$CREATE_TABLE_RESULT" | jq -r '.QueryExecutionId // empty' 2>/
dev/null || echo "$CREATE_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*"' |
    cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from table creation response"
fi
echo "Table creation query ID: $QUERY_ID"

# Wait for table creation to complete
echo "Waiting for table creation to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Table creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Table creation failed with status: $QUERY_STATUS"
    fi
    echo "Table creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))

```

```
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Table creation timed out"
fi

# Verify the table was created
echo "Verifying table creation..."
LIST_TABLE_RESULT=$(aws athena list-table-metadata \
    --catalog-name AwsDataCatalog \
    --database-name "$DATABASE_NAME" \
    --region "$AWS_REGION" 2>&1)
if echo "$LIST_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list tables: $LIST_TABLE_RESULT"
fi
echo "$LIST_TABLE_RESULT"

# Step 3: Query data
echo "Step 3: Running a query on the table..."
QUERY="SELECT os, COUNT(*) count
FROM $DATABASE_NAME.$TABLE_NAME
WHERE date BETWEEN date '2014-07-05' AND date '2014-08-05'
GROUP BY os"

QUERY_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to run query: $QUERY_RESULT"
fi

QUERY_ID=$(echo "$QUERY_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
|| echo "$QUERY_RESULT" | grep -o '"QueryExecutionId": "[^"]*" ' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from query execution response"
fi
echo "Query execution ID: $QUERY_ID"

# Wait for query to complete
echo "Waiting for query to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
```

```
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
    --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Query completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Query failed with status: $QUERY_STATUS"
    fi
    echo "Query in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Query execution timed out"
fi

# Get query results
echo "Getting query results..."
RESULTS=$(aws athena get-query-results --query-execution-id "$QUERY_ID" --region
"$AWS_REGION" 2>&1)
if echo "$RESULTS" | grep -qi "error\|failed"; then
    handle_error "Failed to get query results: $RESULTS"
fi
echo "$RESULTS"

# Download results from S3
echo "Downloading query results from S3..."
S3_PATH=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
    --query "QueryExecution.ResultConfiguration.OutputLocation" --output text \
    --region "$AWS_REGION" 2>&1)
if echo "$S3_PATH" | grep -qi "error\|failed"; then
    handle_error "Failed to get S3 path for results: $S3_PATH"
fi

if [ -z "$S3_PATH" ] || [ "$S3_PATH" = "None" ]; then
    handle_error "S3 path for query results is empty"
fi

DOWNLOAD_RESULT=$(aws s3 cp "$S3_PATH" "./query-results.csv" 2>&1)
if echo "$DOWNLOAD_RESULT" | grep -qi "error\|failed"; then
```

```

    handle_error "Failed to download query results: $DOWNLOAD_RESULT"
fi

# Security: Secure the downloaded file
chmod 600 "./query-results.csv"
echo "Query results downloaded to query-results.csv (permissions: 600)"

# Step 4: Create a named query
echo "Step 4: Creating a named query..."
NAMED_QUERY_RESULT=$(aws athena create-named-query \
    --name "OS Count Query" \
    --description "Count of operating systems in CloudFront logs" \
    --database "$DATABASE_NAME" \
    --query-string "$QUERY" \
    --region "$AWS_REGION" 2>&1)

if echo "$NAMED_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create named query: $NAMED_QUERY_RESULT"
fi

NAMED_QUERY_ID=$(echo "$NAMED_QUERY_RESULT" | jq -r '.NamedQueryId // empty' 2>/
dev/null || echo "$NAMED_QUERY_RESULT" | grep -o '"NamedQueryId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$NAMED_QUERY_ID" ]; then
    handle_error "Failed to extract Named Query ID from response"
fi
echo "Named query created with ID: $NAMED_QUERY_ID"

# List named queries
echo "Listing named queries..."
LIST_QUERIES_RESULT=$(aws athena list-named-queries --region "$AWS_REGION" 2>&1)
if echo "$LIST_QUERIES_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list named queries: $LIST_QUERIES_RESULT"
fi
echo "$LIST_QUERIES_RESULT"

# Get the named query details
echo "Getting named query details..."
GET_QUERY_RESULT=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --region "$AWS_REGION" 2>&1)
if echo "$GET_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to get named query: $GET_QUERY_RESULT"
fi

```

```
echo "$GET_QUERY_RESULT"

# Execute the named query
echo "Executing the named query..."
QUERY_STRING=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --query "NamedQuery.QueryString" --output text --region "$AWS_REGION" 2>&1)
if echo "$QUERY_STRING" | grep -qi "error\\|failed"; then
    handle_error "Failed to get query string: $QUERY_STRING"
fi

if [ -z "$QUERY_STRING" ] || [ "$QUERY_STRING" = "None" ]; then
    handle_error "Query string is empty"
fi

EXEC_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY_STRING" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$EXEC_RESULT" | grep -qi "error\\|failed"; then
    handle_error "Failed to execute named query: $EXEC_RESULT"
fi

QUERY_ID=$(echo "$EXEC_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
    || echo "$EXEC_RESULT" | grep -o '"QueryExecutionId": "[^"]*"' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from named query execution response"
fi
echo "Named query execution ID: $QUERY_ID"

# Wait for named query to complete
echo "Waiting for named query execution to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
        "$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
        "$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Named query execution completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
    then
        handle_error "Named query execution failed with status: $QUERY_STATUS"
    fi
done
```

```

    fi
    echo "Named query execution in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Named query execution timed out"
fi

# Summary of resources created
echo ""
echo "=====
echo "RESOURCES CREATED"
echo "=====
echo "- S3 Bucket: $S3_BUCKET"
echo "- Database: $DATABASE_NAME"
echo "- Table: $TABLE_NAME"
echo "- Named Query: $NAMED_QUERY_ID"
echo "- Query results saved to: query-results.csv"
echo "=====

# Auto-confirm cleanup
echo ""
echo "=====
echo "CLEANUP CONFIRMATION"
echo "=====
echo "Starting cleanup..."
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup..."

    # Delete named query
    echo "Deleting named query: $NAMED_QUERY_ID"
    DELETE_QUERY_RESULT=$(aws athena delete-named-query --named-query-id
"$NAMED_QUERY_ID" \
    --region "$AWS_REGION" 2>&1)
    if echo "$DELETE_QUERY_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to delete named query: $DELETE_QUERY_RESULT"
    else
        echo "Named query deleted successfully."
    fi
fi

```

```

# Drop table
echo "Dropping table: $TABLE_NAME"
DROP_TABLE_RESULT=$(aws athena start-query-execution \
    --query-string "DROP TABLE IF EXISTS $DATABASE_NAME.$TABLE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$DROP_TABLE_RESULT" | grep -qi "error\|failed"; then
    echo "Warning: Failed to drop table: $DROP_TABLE_RESULT"
else
    QUERY_ID=$(echo "$DROP_TABLE_RESULT" | jq -r '.QueryExecutionId // empty'
2>/dev/null || echo "$DROP_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*'
| cut -d'"' -f4)
    if [ -n "$QUERY_ID" ]; then
        echo "Waiting for table deletion to complete..."

        ELAPSED=0
        while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
            QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
            if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                echo "Table dropped successfully."
                break
            elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
"CANCELLED" ]; then
                echo "Warning: Table deletion failed with status:
$QUERY_STATUS"
                break
            fi
            echo "Table deletion in progress, status: $QUERY_STATUS"
            sleep 2
            ((ELAPSED+=2))
        done
    fi
fi

# Drop database
echo "Dropping database: $DATABASE_NAME"
DROP_DB_RESULT=$(aws athena start-query-execution \
    --query-string "DROP DATABASE IF EXISTS $DATABASE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

```



```

    if echo "$DROP_DB_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop database: $DROP_DB_RESULT"
    else
        QUERY_ID=$(echo "$DROP_DB_RESULT" | jq -r '.QueryExecutionId // empty'
2>/dev/null || echo "$DROP_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' |
cut -d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for database deletion to complete..."

            ELAPSED=0
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Database dropped successfully."
                    break
                elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
"CANCELLED" ]; then
                    echo "Warning: Database deletion failed with status:
$QUERY_STATUS"
                    break
                fi
                echo "Database deletion in progress, status: $QUERY_STATUS"
                sleep 2
                ((ELAPSED+=2))
            done
        fi
    fi

    # Empty and delete S3 bucket (only if not shared)
    if [ "$BUCKET_IS_SHARED" = false ]; then
        echo "Emptying S3 bucket: $S3_BUCKET"
        EMPTY_BUCKET_RESULT=$(aws s3 rm "s3://$S3_BUCKET" --recursive 2>&1)
        if echo "$EMPTY_BUCKET_RESULT" | grep -qi "error\|failed"; then
            echo "Warning: Failed to empty S3 bucket: $EMPTY_BUCKET_RESULT"
        else
            echo "S3 bucket emptied successfully."
        fi

        echo "Deleting S3 bucket: $S3_BUCKET"
        DELETE_BUCKET_RESULT=$(aws s3 rb "s3://$S3_BUCKET" 2>&1)
    fi

```

```
if echo "$DELETE_BUCKET_RESULT" | grep -qi "error\|failed"; then
    echo "Warning: Failed to delete S3 bucket: $DELETE_BUCKET_RESULT"
else
    echo "S3 bucket deleted successfully."
fi
else
    echo "Skipping S3 bucket deletion (shared resource)"
fi

# Security: Remove downloaded query results
if [ -f "./query-results.csv" ]; then
    if command -v shred &>/dev/null; then
        shred -vfz -n 3 "./query-results.csv" 2>/dev/null || rm -f "./query-
results.csv"
    else
        rm -f "./query-results.csv"
    fi
    echo "Query results file securely removed."
fi

    echo "Cleanup completed."
fi

echo "Tutorial completed successfully!"
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dei comandi della AWS CLI .
 - [Cap](#)
 - [CreateNamedQuery](#)
 - [DeleteNamedQuery](#)
 - [GetNamedQuery](#)
 - [GetQueryExecution](#)
 - [GetQueryResults](#)
 - [ListDatabases](#)
 - [ListNamedQueries](#)
 - [ListTableMetadata](#)
 - [Mb](#)
 - [Rb](#)

- [Rm](#)
- [StartQueryExecution](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Lavora con le funzionalità di blocco degli oggetti di Amazon S3 utilizzando un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare le funzionalità Object Lock di S3.

.NET

SDK per .NET

Note

C'è altro su [GitHub](#). Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegue uno scenario interattivo che dimostra le funzionalità Object Lock di Amazon S3.

```
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ObjectLockScenario;

public static class S3ObjectLockWorkflow
{
    /*
     Before running this .NET code example, set up your development environment,
     including your credentials.
    */
}
```

This .NET example performs the following tasks:

1. Create test Amazon Simple Storage Service (S3) buckets with different lock policies.
2. Upload sample objects to each bucket.
3. Set some Legal Hold and Retention Periods on objects and buckets.
4. Investigate lock policies by viewing settings or attempting to delete or overwrite objects.
5. Clean up objects and buckets.

*/

```
public static S3ActionsWrapper _s3ActionsWrapper = null!;
public static IConfiguration _configuration = null!;
private static string _resourcePrefix = null!;
private static string noLockBucketName = null!;
private static string lockEnabledBucketName = null!;
private static string retentionAfterCreationBucketName = null!;
private static List<string> bucketNames = new List<string>();
private static List<string> fileNames = new List<string>();

public static async Task Main(string[] args)
{
    // Set up dependency injection for the Amazon service.
    using var host = Host.CreateDefaultBuilder(args)
        .ConfigureLogging(logging =>
            logging.AddFilter("System", LogLevel.Debug)
                .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
        .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonS3>()
                .AddTransient<S3ActionsWrapper>()
        )
        .Build();

    _configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("settings.json") // Load settings from .json file.
        .AddJsonFile("settings.local.json",
            true) // Optionally, load local settings.
        .Build();

    ConfigurationSetup();
```

```

        ServicesSetup(host);

    try
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)
Object Locking Feature Scenario.");
        Console.WriteLine(new string('-', 80));
        await Setup(true);

        await DemoActionChoices();

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Cleaning up resources.");
        Console.WriteLine(new string('-', 80));
        await Cleanup(true);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Amazon S3 Object Locking Scenario is complete.");
        Console.WriteLine(new string('-', 80));
    }
    catch (Exception ex)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"There was a problem: {ex.Message}");
        await Cleanup(true);
        Console.WriteLine(new string('-', 80));
    }
}

/// <summary>
/// Populate the services for use within the console application.
/// </summary>
/// <param name="host">The services host.</param>
private static void ServicesSetup(IHost host)
{
    _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
}

/// <summary>
/// Any setup operations needed.
/// </summary>
public static void ConfigurationSetup()

```

```
{
    _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

    noLockBucketName = _resourcePrefix + "-no-lock";
    lockEnabledBucketName = _resourcePrefix + "-lock-enabled";
    retentionAfterCreationBucketName = _resourcePrefix + "-retention-after-creation";

    bucketNames.Add(noLockBucketName);
    bucketNames.Add(lockEnabledBucketName);
    bucketNames.Add(retentionAfterCreationBucketName);
}

// <summary>
/// Deploy necessary resources for the scenario.
/// </summary>
/// <param name="interactive">True to run as interactive.</param>
/// <returns>True if successful.</returns>
public static async Task<bool> Setup(bool interactive)
{
    Console.WriteLine(
        "\nFor this scenario, we will use the AWS SDK for .NET to create
several S3\n" +
        "buckets and files to demonstrate working with S3 locking features.
\n");

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Press Enter when you are ready to start.");
    if (interactive)
        Console.ReadLine();

    Console.WriteLine("\nS3 buckets can be created either with or without
object lock enabled.");
    await _s3ActionsWrapper.CreateBucketWithObjectLock(noLockBucketName,
false);
    await _s3ActionsWrapper.CreateBucketWithObjectLock(lockEnabledBucketName,
true);
    await
_s3ActionsWrapper.CreateBucketWithObjectLock(retentionAfterCreationBucketName,
false);

    Console.WriteLine("Press Enter to continue.");
    if (interactive)
        Console.ReadLine();
}
```

```
        Console.WriteLine("\nA bucket can be configured to use object locking  
with a default retention period.");  
        await  
_s3ActionsWrapper.ModifyBucketDefaultRetention(retentionAfterCreationBucketName,  
true,  
        ObjectLockRetentionMode.Governance, DateTime.UtcNow.AddDays(1));  
  
        Console.WriteLine("Press Enter to continue.");  
        if (interactive)  
            Console.ReadLine();  
  
        Console.WriteLine("\nObject lock policies can also be added to existing  
buckets.");  
        await _s3ActionsWrapper.EnableObjectLockOnBucket(lockEnabledBucketName);  
  
        Console.WriteLine("Press Enter to continue.");  
        if (interactive)  
            Console.ReadLine();  
  
        // Upload some files to the buckets.  
        Console.WriteLine("\nNow let's add some test files:");  
        var fileName = _configuration["exampleFileName"] ?? "exampleFile.txt";  
        int fileCount = 2;  
        // Create the file if it does not already exist.  
        if (!File.Exists(fileName))  
        {  
            await using StreamWriter sw = File.CreateText(fileName);  
            await sw.WriteLineAsync(  
                "This is a sample file for uploading to a bucket.");  
        }  
  
        foreach (var bucketName in bucketNames)  
        {  
            for (int i = 0; i < fileCount; i++)  
            {  
                var numberedFileName = Path.GetFileNameWithoutExtension(fileName)  
+ i + Path.GetExtension(fileName);  
                fileNames.Add(numberedFileName);  
                await _s3ActionsWrapper.UploadFileAsync(bucketName,  
numberedFileName, fileName);  
            }  
        }  
        Console.WriteLine("Press Enter to continue.");
```

```

        if (interactive)
            Console.ReadLine();

        if (!interactive)
            return true;
        Console.WriteLine("\nNow we can set some object lock policies on
individual files:");
        foreach (var bucketName in bucketNames)
        {
            for (int i = 0; i < fileNames.Count; i++)
            {
                // No modifications to the objects in the first bucket.
                if (bucketName != bucketNames[0])
                {
                    var exampleFileName = fileNames[i];
                    switch (i)
                    {
                        case 0:
                            {
                                var question =
                                    $"Would you like to add a legal hold to
{exampleFileName} in {bucketName}? (y/n)";
                                if (GetYesNoResponse(question))
                                {
                                    // Set a legal hold.
                                    await
_s3ActionsWrapper.ModifyObjectLegalHold(bucketName, exampleFileName,
ObjectLockLegalHoldStatus.On);

                                }
                                break;
                            }
                        case 1:
                            {
                                var question =
                                    $"Would you like to add a 1 day Governance
retention period to {exampleFileName} in {bucketName}? (y/n)" +
                                    "\nReminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.";
                                if (GetYesNoResponse(question))
                                {
                                    // Set a Governance mode retention period for
1 day.

```



```

                                await
_s3ActionsWrapper.ModifyObjectRetentionPeriod(
                                bucketName, exampleFileName,
                                ObjectLockRetentionMode.Governance,
                                DateTime.UtcNow.AddDays(1));
                                }
                                break;
                                }
                                }
                                }
                                }
                                Console.WriteLine(new string('-', 80));
                                return true;
                                }

// <summary>
/// List all of the current buckets and objects.
/// </summary>
/// <param name="interactive">True to run as interactive.</param>
/// <returns>The list of buckets and objects.</returns>
public static async Task<List<S3ObjectVersion>> ListBucketsAndObjects(bool
interactive)
{
    var allObjects = new List<S3ObjectVersion>();
    foreach (var bucketName in bucketNames)
    {
        var objectsInBucket = await
_s3ActionsWrapper.ListBucketObjectsAndVersions(bucketName);
        foreach (var objectKey in objectsInBucket.Versions)
        {
            allObjects.Add(objectKey);
        }
    }

    if (interactive)
    {
        Console.WriteLine("\nCurrent buckets and objects:\n");
        int i = 0;
        foreach (var bucketObject in allObjects)
        {
            i++;
            Console.WriteLine(

```

```

        $"{i}: {bucketObject.Key} \n\tBucket:
{bucketObject.BucketName}\n\tVersion: {bucketObject.VersionId}");
    }
}

return allObjects;
}

/// <summary>
/// Present the user with the demo action choices.
/// </summary>
/// <returns>Async task.</returns>
public static async Task<bool> DemoActionChoices()
{
    var choices = new string[]{
        "List all files in buckets.",
        "Attempt to delete a file.",
        "Attempt to delete a file with retention period bypass.",
        "Attempt to overwrite a file.",
        "View the object and bucket retention settings for a file.",
        "View the legal hold settings for a file.",
        "Finish the scenario."};

    var choice = 0;
    // Keep asking the user until they choose to move on.
    while (choice != 6)
    {
        Console.WriteLine(new string('-', 80));
        choice = GetChoiceResponse(
            "\nExplore the S3 locking features by selecting one of the
following choices:"
            , choices);
        Console.WriteLine(new string('-', 80));
        switch (choice)
        {
            case 0:
            {
                await ListBucketsAndObjects(true);
                break;
            }
            case 1:
            {
                Console.WriteLine("\nEnter the number of the object to
delete:");

```

```

        var allFiles = await ListBucketsAndObjects(true);
        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, false, allFiles[fileChoice].VersionId);
        break;
    }
    case 2:
    {
        Console.WriteLine("\nEnter the number of the object to
delete:");

        var allFiles = await ListBucketsAndObjects(true);
        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, true, allFiles[fileChoice].VersionId);
        break;
    }
    case 3:
    {
        var allFiles = await ListBucketsAndObjects(true);
        Console.WriteLine("\nEnter the number of the object to
overwrite:");

        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        // Create the file if it does not already exist.
        if (!File.Exists(allFiles[fileChoice].Key))
        {
            await using StreamWriter sw =
File.CreateText(allFiles[fileChoice].Key);
            await sw.WriteLineAsync(
                "This is a sample file for uploading to a
bucket.");
        }
        await
_s3ActionsWrapper.UploadFileAsync(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, allFiles[fileChoice].Key);
        break;
    }
    case 4:
    {
        var allFiles = await ListBucketsAndObjects(true);

```

```

        Console.WriteLine("\nEnter the number of the object and
bucket to view:");
        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        await
_s3ActionsWrapper.GetObjectRetention(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
        await
_s3ActionsWrapper.GetBucketObjectLockConfiguration(allFiles[fileChoice].BucketName);
        break;
    }
    case 5:
    {
        var allFiles = await ListBucketsAndObjects(true);
        Console.WriteLine("\nEnter the number of the object to
view:");
        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        await
_s3ActionsWrapper.GetObjectLegalHold(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
        break;
    }
    }
    }
    return true;
}

// <summary>
/// Clean up the resources from the scenario.
/// </summary>
/// <param name="interactive">True to run as interactive.</param>
/// <returns>True if successful.</returns>
public static async Task<bool> Cleanup(bool interactive)
{
    Console.WriteLine(new string('-', 80));

    if (!interactive || GetYesNoResponse("Do you want to clean up all files
and buckets? (y/n) "))
    {
        // Remove all locks and delete all buckets and objects.
        var allFiles = await ListBucketsAndObjects(false);
        foreach (var fileInfo in allFiles)
        {

```

```
        // Check for a legal hold.
        var legalHold = await
        _s3ActionsWrapper.GetObjectLegalHold(fileInfo.BucketName, fileInfo.Key);
        if (legalHold?.Status?.Value == ObjectLockLegalHoldStatus.On)
        {
            await
            _s3ActionsWrapper.ModifyObjectLegalHold(fileInfo.BucketName, fileInfo.Key,
            ObjectLockLegalHoldStatus.Off);
        }

        // Check for a retention period.
        var retention = await
        _s3ActionsWrapper.GetObjectRetention(fileInfo.BucketName, fileInfo.Key);
        var hasRetentionPeriod = retention?.Mode ==
        ObjectLockRetentionMode.Governance && retention.RetainUntilDate >
        DateTime.UtcNow.Date;
        await
        _s3ActionsWrapper.DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key,
        hasRetentionPeriod, fileInfo.VersionId);
    }

    foreach (var bucketName in bucketNames)
    {
        await _s3ActionsWrapper.DeleteBucketByName(bucketName);
    }
}
else
{
    Console.WriteLine(
        "Ok, we'll leave the resources intact.\n" +
        "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
    );
}

Console.WriteLine(new string('-', 80));
return true;
}

/// <summary>
/// Helper method to get a yes or no response from the user.
/// </summary>
```

```
    /// <param name="question">The question string to print on the console.</  
param>  
    /// <returns>True if the user responds with a yes.</returns>  
    private static bool GetYesNoResponse(string question)  
    {  
        Console.WriteLine(question);  
        var ynResponse = Console.ReadLine();  
        var response = ynResponse != null && ynResponse.Equals("y",  
StringComparison.InvariantCultureIgnoreCase);  
        return response;  
    }  
  
    /// <summary>  
    /// Helper method to get a choice response from the user.  
    /// </summary>  
    /// <param name="question">The question string to print on the console.</  
param>  
    /// <param name="choices">The choices to print on the console.</param>  
    /// <returns>The index of the selected choice</returns>  
    private static int GetChoiceResponse(string? question, string[] choices)  
    {  
        if (question != null)  
        {  
            Console.WriteLine(question);  
  
            for (int i = 0; i < choices.Length; i++)  
            {  
                Console.WriteLine($"{i + 1}. {choices[i]}");  
            }  
        }  
  
        var choiceNumber = 0;  
        while (choiceNumber < 1 || choiceNumber > choices.Length)  
        {  
            var choice = Console.ReadLine();  
            Int32.TryParse(choice, out choiceNumber);  
        }  
  
        return choiceNumber - 1;  
    }  
}
```

Una classe wrapper per le funzioni S3.

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

namespace S3ObjectLockScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, IConfiguration configuration)
    {
        _amazonS3 = amazonS3;
    }

    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the
    bucket.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
    enableObjectLock)
    {
        Console.WriteLine($"\\tCreating bucket {bucketName} with object lock
    {enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
```

```

        ObjectLockEnabledForBucket = enableObjectLock,
    };

    var response = await _amazonS3.PutBucketAsync(request);

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error creating bucket: '{ex.Message}'");
    return false;
}
}

/// <summary>
/// Enable object lock on an existing bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket to modify.</param>
/// <returns>True if successful.</returns>
public async Task<bool> EnableObjectLockOnBucket(string bucketName)
{
    try
    {
        // First, enable Versioning on the bucket.
        await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
        {
            BucketName = bucketName,
            VersioningConfig = new S3BucketVersioningConfig()
            {
                EnableMfaDelete = false,
                Status = VersionStatus.Enabled
            }
        });

        var request = new PutObjectLockConfigurationRequest()
        {
            BucketName = bucketName,
            ObjectLockConfiguration = new ObjectLockConfiguration()
            {
                ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
            },
        };
    }
}

```



```

        var response = await
        _amazonS3.PutObjectLockConfigurationAsync(request);
        Console.WriteLine($"{bucketName}\tAdded an object lock policy to bucket
{bucketName}");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
        return false;
    }
}

/// <summary>
/// Set or modify a retention period on an object in an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The key of the object.</param>
/// <param name="retention">The retention mode.</param>
/// <param name="retainUntilDate">The date retention expires.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
    string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
{
    try
    {
        var request = new PutObjectRetentionRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
            Retention = new ObjectLockRetention()
            {
                Mode = retention,
                RetainUntilDate = retainUntilDate
            }
        };

        var response = await _amazonS3.PutObjectRetentionAsync(request);
        Console.WriteLine($"{objectKey}\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)

```

```

        {
            Console.WriteLine($"\\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
    {
        var enabledString = enableObjectLock ? "Enabled" : "Disabled";
        var timeDifference = retainUntilDate.Subtract(DateTime.Now);
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                    Rule = new ObjectLockRule()
                    {
                        DefaultRetention = new DefaultRetention()
                        {
                            Mode = retention,

```

```

        Days = timeDifference.Days // Can be specified in
days or years but not both.
    }
}
};

var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
Console.WriteLine($"\\tAdded a default retention to bucket
{bucketName}.");
return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"\\tError modifying object lock: '{ex.Message}'");
    return false;
}
}

/// <summary>
/// Get the retention period for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object retention details.</returns>
public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
    string objectKey)
{
    try
    {
        var request = new GetObjectRetentionRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };

        var response = await _amazonS3.GetObjectRetentionAsync(request);
        Console.WriteLine($"\\tObject retention for {objectKey} in
{bucketName}: " +
            $"\\n\\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
        return response.Retention;
    }
}

```

```

        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tUnable to fetch object lock retention:
'{ex.Message}''");
            return new ObjectLockRetention();
        }
    }

    /// <summary>
    /// Set or modify a legal hold on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="holdStatus">The On or Off status for the legal hold.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectLegalHold(string bucketName,
        string objectKey, ObjectLockLegalHoldStatus holdStatus)
    {
        try
        {
            var request = new PutObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                LegalHold = new ObjectLockLegalHold()
                {
                    Status = holdStatus
                }
            };

            var response = await _amazonS3.PutObjectLegalHoldAsync(request);
            Console.WriteLine($"\\tModified legal hold for {objectKey} in
{bucketName}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tError modifying legal hold: '{ex.Message}''");
            return false;
        }
    }

    /// <summary>
    /// Get the legal hold details for an S3 object.

```

```
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object legal hold details.</returns>
public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
    string objectKey)
{
    try
    {
        var request = new GetObjectLegalHoldRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };

        var response = await _amazonS3.GetObjectLegalHoldAsync(request);
        Console.WriteLine($"{objectKey} legal hold for {objectKey} in
{bucketName}: " +
            $"{response.LegalHold.Status}");
        return response.LegalHold;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Unable to fetch legal hold: '{ex.Message}'");
        return new ObjectLockLegalHold();
    }
}

/// <summary>
/// Get the object lock configuration details for an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket to get details.</param>
/// <returns>The bucket's object lock configuration details.</returns>
public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
{
    try
    {
        var request = new GetObjectLockConfigurationRequest()
        {
            BucketName = bucketName
        };
    }
```

```

        var response = await
        _amazonS3.GetObjectLockConfigurationAsync(request);
        Console.WriteLine($"\\tBucket object lock config for {bucketName} in
{bucketName}: " +
                        $"\\n\\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                        $"\\n\\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

        return response.ObjectLockConfiguration;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to fetch object lock config:
'{ex.Message}');
        return new ObjectLockConfiguration();
    }
}

/// <summary>
/// Upload a file from the local computer to an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <param name="objectName">The object to upload.</param>
/// <param name="filePath">The path, including file name, of the object to
upload.</param>
/// <returns>True if success.</returns>
public async Task<bool> UploadFileAsync(string bucketName, string objectName,
string filePath)
{
    var request = new PutObjectRequest
    {
        BucketName = bucketName,
        Key = objectName,
        FilePath = filePath,
        ChecksumAlgorithm = ChecksumAlgorithm.SHA256
    };

    var response = await _amazonS3.PutObjectAsync(request);
    if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
    {
        Console.WriteLine($"\\tSuccessfully uploaded {objectName} to
{bucketName}.");
        return true;
    }
}

```

```
    }
    else
    {
        Console.WriteLine($"\\tCould not upload {objectName} to
{bucketName}.");
        return false;
    }
}

/// <summary>
/// List bucket objects and versions.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <returns>The list of objects and versions.</returns>
public async Task<ListVersionsResponse> ListBucketObjectsAndVersions(string
bucketName)
{
    var request = new ListVersionsRequest()
    {
        BucketName = bucketName
    };

    var response = await _amazonS3.ListVersionsAsync(request);
    return response;
}

/// <summary>
/// Delete an object from a specific bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <param name="objectKey">The key of the object to delete.</param>
/// <param name="hasRetention">True if the object has retention settings.</
param>
/// <param name="versionId">Optional versionId.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteObjectFromBucket(string bucketName, string
objectKey, bool hasRetention, string? versionId = null)
{
    try
    {
        var request = new DeleteObjectRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
```

```
        VersionId = versionId,
    };
    if (hasRetention)
    {
        // Set the BypassGovernanceRetention header
        // if the file has retention settings.
        request.BypassGovernanceRetention = true;
    }
    await _amazonS3.DeleteObjectAsync(request);
    Console.WriteLine(
        $"Deleted {objectKey} in {bucketName}.");
    return true;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Unable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
    return false;
}
}

/// <summary>
/// Delete a specific bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <param name="objectKey">The key of the object to delete.</param>
/// <param name="versionId">Optional versionId.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteBucketByName(string bucketName)
{
    try
    {
        var request = new DeleteBucketRequest() { BucketName = bucketName, };
        var response = await _amazonS3.DeleteBucketAsync(request);
        Console.WriteLine($"Delete for {bucketName} complete.");
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Unable to delete bucket {bucketName}: " +
ex.Message);
        return false;
    }
}
```



```
}  
  
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per .NET .
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

Go

SDK per Go V2

 Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegue uno scenario interattivo che dimostra le funzionalità Object Lock di Amazon S3.

```
import (  
    "context"  
    "fmt"  
    "log"  
    "strings"  
  
    "s3_object_lock/actions"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
```

```

    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// ObjectLockScenario contains the steps to run the S3 Object Lock workflow.
type ObjectLockScenario struct {
    questioner demotools.IQuestioner
    resources  Resources
    s3Actions  *actions.S3Actions
    sdkConfig  aws.Config
}

// NewObjectLockScenario constructs a new ObjectLockScenario instance.
func NewObjectLockScenario(sdkConfig aws.Config, questioner
    demotools.IQuestioner) ObjectLockScenario {
    scenario := ObjectLockScenario{
        questioner: questioner,
        resources:  Resources{},
        s3Actions:  &actions.S3Actions{S3Client: s3.NewFromConfig(sdkConfig)},
        sdkConfig:  sdkConfig,
    }
    scenario.s3Actions.S3Manager = manager.NewUploader(scenario.s3Actions.S3Client)
    scenario.resources.init(scenario.s3Actions, questioner)
    return scenario
}

type nameLocked struct {
    name  string
    locked bool
}

var createInfo = []nameLocked{
    {"standard-bucket", false},
    {"lock-bucket", true},
    {"retention-bucket", false},
}

// CreateBuckets creates the S3 buckets required for the workflow.
func (scenario *ObjectLockScenario) CreateBuckets(ctx context.Context) {
    log.Println("Let's create some S3 buckets to use for this workflow.")
    success := false
    for !success {
        prefix := scenario.questioner.Ask(
            "This example creates three buckets. Enter a prefix to name your buckets
            (remember bucket names must be globally unique):")
    }
}

```

```
for _, info := range createInfo {
    log.Println(fmt.Sprintf("%s.%s", prefix, info.name))
    bucketName, err := scenario.s3Actions.CreateBucketWithLock(ctx,
fmt.Sprintf("%s.%s", prefix, info.name), scenario.sdkConfig.Region, info.locked)
    if err != nil {
        switch err.(type) {
            case *types.BucketAlreadyExists, *types.BucketAlreadyOwnedByYou:
                log.Printf("Couldn't create bucket %s.\n", bucketName)
            default:
                panic(err)
        }
        break
    }
    scenario.resources.demoBuckets[info.name] = &DemoBucket{
        name:      bucketName,
        objectKeys: []string{},
    }
    log.Printf("Created bucket %s.\n", bucketName)
}

if len(scenario.resources.demoBuckets) < len(createInfo) {
    scenario.resources.deleteBuckets(ctx)
} else {
    success = true
}
}

log.Println("S3 buckets created.")
log.Println(strings.Repeat("-", 88))
}

// EnableLockOnBucket enables object locking on an existing bucket.
func (scenario *ObjectLockScenario) EnableLockOnBucket(ctx context.Context) {
    log.Println("\nA bucket can be configured to use object locking.")
    scenario.questioner.Ask("Press Enter to continue.")

    var err error
    bucket := scenario.resources.demoBuckets["retention-bucket"]
    err = scenario.s3Actions.EnableObjectLockOnBucket(ctx, bucket.name)
    if err != nil {
        switch err.(type) {
            case *types.NoSuchBucket:
                log.Printf("Couldn't enable object locking on bucket %s.\n", bucket.name)
```

```
        default:
            panic(err)
        }
    } else {
        log.Printf("Object locking enabled on bucket %s.", bucket.name)
    }

    log.Println(strings.Repeat("-", 88))
}

// SetDefaultRetentionPolicy sets a default retention governance policy on a
// bucket.
func (scenario *ObjectLockScenario) SetDefaultRetentionPolicy(ctx
context.Context) {
    log.Println("\nA bucket can be configured to use object locking with a default
retention period.")

    bucket := scenario.resources.demoBuckets["retention-bucket"]
    retentionPeriod := scenario.questioner.AskInt("Enter the default retention
period in days: ")
    err := scenario.s3Actions.ModifyDefaultBucketRetention(ctx,
bucket.name, types.ObjectLockEnabledEnabled, int32(retentionPeriod),
types.ObjectLockRetentionModeGovernance)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchBucket:
            log.Printf("Couldn't configure a default retention period on bucket %s.\n",
bucket.name)
        default:
            panic(err)
        }
    } else {
        log.Printf("Default retention policy set on bucket %s with %d day retention
period.", bucket.name, retentionPeriod)
        bucket.retentionEnabled = true
    }

    log.Println(strings.Repeat("-", 88))
}

// UploadTestObjects uploads test objects to the S3 buckets.
func (scenario *ObjectLockScenario) UploadTestObjects(ctx context.Context) {
    log.Println("Uploading test objects to S3 buckets.")
}
```

```

for _, info := range createInfo {
    bucket := scenario.resources.demoBuckets[info.name]
    for i := 0; i < 2; i++ {
        key, err := scenario.s3Actions.UploadObject(ctx, bucket.name,
fmt.Sprintf("example-%d", i),
        fmt.Sprintf("Example object content #%d in bucket %s.", i, bucket.name))
        if err != nil {
            switch err.(type) {
            case *types.NoSuchBucket:
                log.Printf("Couldn't upload %s to bucket %s.\n", key, bucket.name)
            default:
                panic(err)
            }
        } else {
            log.Printf("Uploaded %s to bucket %s.\n", key, bucket.name)
            bucket.objectKeys = append(bucket.objectKeys, key)
        }
    }
}

scenario.questioner.Ask("Test objects uploaded. Press Enter to continue.")
log.Println(strings.Repeat("-", 88))
}

// SetObjectLockConfigurations sets object lock configurations on the test
// objects.
func (scenario *ObjectLockScenario) SetObjectLockConfigurations(ctx
context.Context) {
    log.Println("Now let's set object lock configurations on individual objects.")

    buckets := []*DemoBucket{scenario.resources.demoBuckets["lock-bucket"],
scenario.resources.demoBuckets["retention-bucket"]}
    for _, bucket := range buckets {
        for index, objKey := range bucket.objectKeys {
            switch index {
            case 0:
                if scenario.questioner.AskBool(fmt.Sprintf("\nDo you want to add a legal hold
to %s in %s (y/n)? ", objKey, bucket.name), "y") {
                    err := scenario.s3Actions.PutObjectLegalHold(ctx, bucket.name, objKey, "",
types.ObjectLockLegalHoldStatusOn)
                    if err != nil {
                        switch err.(type) {
                        case *types.NoSuchKey:
                            log.Printf("Couldn't set legal hold on %s.\n", objKey)

```

```

        default:
            panic(err)
        }
    } else {
        log.Printf("Legal hold set on %s.\n", objKey)
    }
}

case 1:
    q := fmt.Sprintf("\nDo you want to add a 1 day Governance retention period to
%s in %s?\n"+
        "Reminder: Only a user with the s3:BypassGovernanceRetention permission is
able to delete this object\n"+
        "or its bucket until the retention period has expired. (y/n) ", objKey,
bucket.name)
    if scenario.questioner.AskBool(q, "y") {
        err := scenario.s3Actions.PutObjectRetention(ctx, bucket.name, objKey,
types.ObjectLockRetentionModeGovernance, 1)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchKey:
                log.Printf("Couldn't set retention period on %s in %s.\n", objKey,
bucket.name)
            default:
                panic(err)
            }
        } else {
            log.Printf("Retention period set to 1 for %s.", objKey)
            bucket.retentionEnabled = true
        }
    }
}
}
}
log.Println(strings.Repeat("-", 88))
}

const (
    ListAll = iota
    DeleteObject
    DeleteRetentionObject
    OverwriteObject
    ViewRetention
    ViewLegalHold
    Finish

```

```
)

// InteractWithObjects allows the user to interact with the objects and test the
// object lock configurations.
func (scenario *ObjectLockScenario) InteractWithObjects(ctx context.Context) {
    log.Println("Now you can interact with the objects to explore the object lock
    configurations.")
    interactiveChoices := []string{
        "List all objects and buckets.",
        "Attempt to delete an object.",
        "Attempt to delete an object with retention period bypass.",
        "Attempt to overwrite a file.",
        "View the retention settings for an object.",
        "View the legal hold settings for an object.",
        "Finish the workflow."}

    choice := ListAll
    for choice != Finish {
        objList := scenario.GetAllObjects(ctx)
        objChoices := scenario.makeObjectChoiceList(objList)
        choice = scenario.questioner.AskChoice("Choose an action from the menu:\n",
        interactiveChoices)
        switch choice {
        case ListAll:
            log.Println("The current objects in the example buckets are:")
            for _, objChoice := range objChoices {
                log.Println("\t", objChoice)
            }
        case DeleteObject, DeleteRetentionObject:
            objChoice := scenario.questioner.AskChoice("Enter the number of the object to
            delete:\n", objChoices)
            obj := objList[objChoice]
            deleted, err := scenario.s3Actions.DeleteObject(ctx, obj.bucket, obj.key,
            obj.versionId, choice == DeleteRetentionObject)
            if err != nil {
                switch err.(type) {
                case *types.NoSuchKey:
                    log.Println("Nothing to delete.")
                default:
                    panic(err)
                }
            } else if deleted {
                log.Printf("Object %s deleted.\n", obj.key)
            }
        }
    }
}
```

```
case OverwriteObject:
    objChoice := scenario.questioner.AskChoice("Enter the number of the object to
overwrite:\n", objChoices)
    obj := objList[objChoice]
    _, err := scenario.s3Actions.UploadObject(ctx, obj.bucket, obj.key,
fmt.Sprintf("New content in object %s.", obj.key))
    if err != nil {
        switch err.(type) {
        case *types.NoSuchBucket:
            log.Println("Couldn't upload to nonexistent bucket.")
        default:
            panic(err)
        }
    } else {
        log.Printf("Uploaded new content to object %s.\n", obj.key)
    }
case ViewRetention:
    objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
    obj := objList[objChoice]
    retention, err := scenario.s3Actions.GetObjectRetention(ctx, obj.bucket,
obj.key)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchKey:
            log.Printf("Can't get retention configuration for %s.\n", obj.key)
        default:
            panic(err)
        }
    } else if retention != nil {
        log.Printf("Object %s has retention mode %s until %v.\n", obj.key,
retention.Mode, retention.RetainUntilDate)
    } else {
        log.Printf("Object %s does not have object retention configured.\n", obj.key)
    }
case ViewLegalHold:
    objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
    obj := objList[objChoice]
    legalHold, err := scenario.s3Actions.GetObjectLegalHold(ctx, obj.bucket,
obj.key, obj.versionId)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchKey:
```



```

    log.Printf("Can't get legal hold configuration for %s.\n", obj.key)
default:
    panic(err)
}
} else if legalHold != nil {
    log.Printf("Object %s has legal hold %v.", obj.key, *legalHold)
} else {
    log.Printf("Object %s does not have legal hold configured.", obj.key)
}
case Finish:
    log.Println("Let's clean up.")
}
log.Println(strings.Repeat("-", 88))
}
}

type BucketKeyVersionId struct {
    bucket    string
    key       string
    versionId string
}

// GetAllObjects gets the object versions in the example S3 buckets and returns
// them in a flattened list.
func (scenario *ObjectLockScenario) GetAllObjects(ctx context.Context)
[]BucketKeyVersionId {
    var objectList []BucketKeyVersionId
    for _, info := range createInfo {
        bucket := scenario.resources.demoBuckets[info.name]
        versions, err := scenario.s3Actions.ListObjectVersions(ctx, bucket.name)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchBucket:
                log.Printf("Couldn't get object versions for %s.\n", bucket.name)
            default:
                panic(err)
            }
        } else {
            for _, version := range versions {
                objectList = append(objectList,
                    BucketKeyVersionId{bucket: bucket.name, key: *version.Key, versionId:
                        *version.VersionId})
            }
        }
    }
}

```

```
}
return objectList
}

// makeObjectChoiceList makes the object version list into a list of strings that
// are displayed
// as choices.
func (scenario *ObjectLockScenario) makeObjectChoiceList(bucketObjects
[]BucketKeyVersionId) []string {
    choices := make([]string, len(bucketObjects))
    for i := 0; i < len(bucketObjects); i++ {
        choices[i] = fmt.Sprintf("%s in %s with VersionId %s.",
            bucketObjects[i].key, bucketObjects[i].bucket, bucketObjects[i].versionId)
    }
    return choices
}

// Run runs the S3 Object Lock scenario.
func (scenario *ObjectLockScenario) Run(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            _, isMock := scenario.questioner.(*demotools.MockQuestioner)
            if isMock || scenario.questioner.AskBool("Do you want to see the full error
message (y/n)?", "y") {
                log.Println(r)
            }
            scenario.resources.Cleanup(ctx)
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Println("Welcome to the Amazon S3 Object Lock Feature Scenario.")
    log.Println(strings.Repeat("-", 88))

    scenario.CreateBuckets(ctx)
    scenario.EnableLockOnBucket(ctx)
    scenario.SetDefaultRetentionPolicy(ctx)
    scenario.UploadTestObjects(ctx)
    scenario.SetObjectLockConfigurations(ctx)
    scenario.InteractWithObjects(ctx)

    scenario.resources.Cleanup(ctx)
}
```

```
log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

Definisce una struttura per il wrapping delle azioni S3 utilizzate in questo esempio.

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// CreateBucketWithLock creates a new S3 bucket with optional object locking
// enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
    region string, enableObjectLock bool) (string, error) {
    input := &s3.CreateBucketInput{
        Bucket: aws.String(bucket),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    },
```

```

}

if enableObjectLock {
    input.ObjectLockEnabledForBucket = aws.Bool(true)
}

_, err := actor.S3Client.CreateBucket(ctx, input)
if err != nil {
    var owned *types.BucketAlreadyOwnedByYou
    var exists *types.BucketAlreadyExists
    if errors.As(err, &owned) {
        log.Printf("You already own bucket %s.\n", bucket)
        err = owned
    } else if errors.As(err, &exists) {
        log.Printf("Bucket %s already exists.\n", bucket)
        err = exists
    }
} else {
    err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
        ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
    }
}

return bucket, err
}

// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
    var status *types.ObjectLockLegalHoldStatus
    input := &s3.GetObjectLegalHoldInput{
        Bucket:    aws.String(bucket),
        Key:       aws.String(key),
        VersionId: aws.String(versionId),
    }

    output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
    if err != nil {
        var noSuchKeyErr *types.NoSuchKey
        var apiErr *smithy.GenericAPIError

```

```

    if errors.As(err, &noSuchKeyErr) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noSuchKeyErr
    } else if errors.As(err, &apiErr) {
        switch apiErr.ErrorCode() {
        case "NoSuchObjectLockConfiguration":
            log.Printf("Object %s does not have an object lock configuration.\n", key)
            err = nil
        case "InvalidRequest":
            log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
            err = nil
        }
    }
} else {
    status = &output.LegalHold.Status
}

return status, err
}

// GetObjectLockConfiguration retrieves the object lock configuration for an S3
// bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
    string) (*types.ObjectLockConfiguration, error) {
    var lockConfig *types.ObjectLockConfiguration
    input := &s3.GetObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
    }

    output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
            "ObjectLockConfigurationNotFoundError" {
            log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
            err = nil
        }
    } else {

```

```
    lockConfig = output.ObjectLockConfiguration
  }

  return lockConfig, err
}

// GetObjectRetention retrieves the object retention configuration for an S3
// object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
string) (*types.ObjectLockRetention, error) {
  var retention *types.ObjectLockRetention
  input := &s3.GetObjectRetentionInput{
    Bucket: aws.String(bucket),
    Key:    aws.String(key),
  }

  output, err := actor.S3Client.GetObjectRetention(ctx, input)
  if err != nil {
    var noKey *types.NoSuchKey
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noKey) {
      log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
      err = noKey
    } else if errors.As(err, &apiErr) {
      switch apiErr.ErrorCode() {
      case "NoSuchObjectLockConfiguration":
        err = nil
      case "InvalidRequest":
        log.Printf("Bucket %s does not have locking enabled.", bucket)
        err = nil
      }
    }
  } else {
    retention = output.Retention
  }

  return retention, err
}

// PutObjectLegalHold sets the legal hold configuration for an S3 object.
```

```

func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
{
    input := &s3.PutObjectLegalHoldInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        LegalHold: &types.ObjectLockLegalHold{
            Status: legalHoldStatus,
        },
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }

    _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}

// ModifyDefaultBucketRetention modifies the default retention period of an
existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
    ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
    retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: lockMode,
            Rule: &types.ObjectLockRule{
                DefaultRetention: &types.DefaultRetention{
                    Days: aws.Int32(retentionPeriod),
                    Mode: retentionMode,
                },
            },
        },
    },
}

```

```

    },
}

_, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
}

return err
}

// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
string) error {
    // Versioning must be enabled on the bucket before object locking is enabled.
    verInput := &s3.PutBucketVersioningInput{
        Bucket: aws.String(bucket),
        VersioningConfiguration: &types.VersioningConfiguration{
            MFADelete: types.MFADeleteDisabled,
            Status:    types.BucketVersioningStatusEnabled,
        },
    }
    _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
        return err
    }

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: types.ObjectLockEnabledEnabled,
        },
    }
    _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)

```



```
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
}

return err
}

// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
error {
    input := &s3.PutObjectRetentionInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        Retention: &types.ObjectLockRetention{
            Mode:          retentionMode,
            RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
        },
        BypassGovernanceRetention: aws.Bool(true),
    }

    _, err := actor.S3Client.PutObjectRetention(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}

// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
string, contents string) (string, error) {
```

```
var outKey string
input := &s3.PutObjectInput{
    Bucket:      aws.String(bucket),
    Key:         aws.String(key),
    Body:        bytes.NewReader([]byte(contents)),
    ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
}
output, err := actor.S3Manager.Upload(ctx, input)
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
} else {
    err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
            bucket)
    } else {
        outKey = *output.Key
    }
}
return outKey, err
}

// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
([]types.ObjectVersion, error) {
    var err error
    var output *s3.ListObjectVersionsOutput
    var versions []types.ObjectVersion
    input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
    versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
    for versionPaginator.HasMorePages() {
        output, err = versionPaginator.NextPage(ctx)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
```

```

    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
}
break
} else {
    versions = append(versions, output.Versions...)
}
}
return versions, err
}

// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
string, versionId string, bypassGovernance bool) (bool, error) {
    deleted := false
    input := &s3.DeleteObjectInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    _, err := actor.S3Client.DeleteObject(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in %s.\n", key, bucket)
            err = noKey
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "AccessDenied":
                log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
                err = nil
            case "InvalidArgument":
                if bypassGovernance {
                    log.Printf("You cannot specify bypass governance on a bucket without lock
enabled.")
                }
                err = nil
            }
        }
    }
}

```

```

    }
    }
    }
} else {
    err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
        ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
        time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.
\n", key, bucket)
    } else {
        deleted = true
    }
}
return deleted, err
}

// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
[]types.ObjectIdentifier, bypassGovernance bool) error {
    if len(objects) == 0 {
        return nil
    }

    input := s3.DeleteObjectsInput{
        Bucket: aws.String(bucket),
        Delete: &types.Delete{
            Objects: objects,
            Quiet:   aws.Bool(true),
        },
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
    if err != nil || len(delOut.Errors) > 0 {
        log.Printf("Error deleting objects from bucket %s.\n", bucket)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucket)
                err = noBucket
            }
        }
    }
}

```

```

    }
} else if len(del0ut.Errors) > 0 {
    for _, outErr := range del0ut.Errors {
        log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
    }
    err = fmt.Errorf("%s", *del0ut.Errors[0].Message)
}
} else {
    for _, del0bjs := range del0ut.Deleted {
        err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: del0bjs.Key},
            time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to be deleted.\n",
                *del0bjs.Key)
        } else {
            log.Printf("Deleted %s.\n", *del0bjs.Key)
        }
    }
}
return err
}

```

Eliminare le risorse.

```

import (
    "context"
    "log"
    "s3_object_lock/actions"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// DemoBucket contains metadata for buckets used in this example.
type DemoBucket struct {

```

```
    name            string
    retentionEnabled bool
    objectKeys       []string
}

// Resources keeps track of AWS resources created during the ObjectLockScenario
// and handles
// cleanup when the scenario finishes.
type Resources struct {
    demoBuckets map[string]*DemoBucket

    s3Actions  *actions.S3Actions
    questioner demotools.IQuestioner
}

// init initializes objects in the Resources struct.
func (resources *Resources) init(s3Actions *actions.S3Actions, questioner
    demotools.IQuestioner) {
    resources.s3Actions = s3Actions
    resources.questioner = questioner
    resources.demoBuckets = map[string]*DemoBucket{}
}

// Cleanup deletes all AWS resources created during the ObjectLockScenario.
func (resources *Resources) Cleanup(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Printf("Something went wrong during cleanup.\n%v\n", r)
            log.Println("Use the AWS Management Console to remove any remaining resources"
                " +
                "that were created for this scenario.")
        }
    }()

    wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
        resources that were created "+
        "during this demo (y/n)?", "y")
    if !wantDelete {
        log.Println("Be sure to remove resources when you're done with them to avoid
            unexpected charges!")
        return
    }

    log.Println("Removing objects from S3 buckets and deleting buckets...")
}
```

```
resources.deleteBuckets(ctx)
//resources.deleteRetentionObjects(resources.retentionBucket,
resources.retentionObjects)

log.Println("Cleanup complete.")
}

// deleteBuckets empties and then deletes all buckets created during the
ObjectLockScenario.
func (resources *Resources) deleteBuckets(ctx context.Context) {
    for _, info := range createInfo {
        bucket := resources.demoBuckets[info.name]
        resources.deleteObjects(ctx, bucket)
        _, err := resources.s3Actions.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
            Bucket: aws.String(bucket.name),
        })
        if err != nil {
            panic(err)
        }
    }
    for _, info := range createInfo {
        bucket := resources.demoBuckets[info.name]
        err := s3.NewBucketNotExistsWaiter(resources.s3Actions.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket.name)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
                bucket.name)
        } else {
            log.Printf("Deleted %s.\n", bucket.name)
        }
    }
    resources.demoBuckets = map[string]*DemoBucket{}
}

// deleteObjects deletes all objects in the specified bucket.
func (resources *Resources) deleteObjects(ctx context.Context, bucket
    *DemoBucket) {
    lockConfig, err := resources.s3Actions.GetObjectLockConfiguration(ctx,
        bucket.name)
    if err != nil {
        panic(err)
    }
    versions, err := resources.s3Actions.ListObjectVersions(ctx, bucket.name)
    if err != nil {
```

```

switch err.(type) {
case *types.NoSuchBucket:
    log.Printf("No objects to get from %s.\n", bucket.name)
default:
    panic(err)
}
}
del0bjects := make([]types.ObjectIdentifier, len(versions))
for i, version := range versions {
    if lockConfig != nil && lockConfig.ObjectLockEnabled ==
types.ObjectLockEnabledEnabled {
        status, err := resources.s3Actions.GetObjectLegalHold(ctx, bucket.name,
*version.Key, *version.VersionId)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchKey:
                log.Printf("Couldn't determine legal hold status for %s in %s.\n",
*version.Key, bucket.name)
            default:
                panic(err)
            }
        } else if status != nil && *status == types.ObjectLockLegalHoldStatusOn {
            err = resources.s3Actions.PutObjectLegalHold(ctx, bucket.name, *version.Key,
*version.VersionId, types.ObjectLockLegalHoldStatusOff)
            if err != nil {
                switch err.(type) {
                case *types.NoSuchKey:
                    log.Printf("Couldn't turn off legal hold for %s in %s.\n", *version.Key,
bucket.name)
                default:
                    panic(err)
                }
            }
        }
    }
    del0bjects[i] = types.ObjectIdentifier{Key: version.Key, VersionId:
version.VersionId}
}
err = resources.s3Actions.DeleteObjects(ctx, bucket.name, del0bjects,
bucket.retentionEnabled)
if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
        log.Println("Nothing to delete.")
    }
}

```



```
default:
    panic(err)
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Go .
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

Java

SDK per Java 2.x

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegue uno scenario interattivo che dimostra le funzionalità Object Lock di Amazon S3.

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

/*
```

Before running this Java V2 code example, set up your development environment, including your credentials.

For more information, see the following documentation topic:

<https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html>

This Java example performs the following tasks:

1. Create test Amazon Simple Storage Service (S3) buckets with different lock policies.
2. Upload sample objects to each bucket.
3. Set some Legal Hold and Retention Periods on objects and buckets.
4. Investigate lock policies by viewing settings or attempting to delete or overwrite objects.
5. Clean up objects and buckets.

```
*/
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0",
"-");
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
    private static final List<String> fileNames = new ArrayList<>();

    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <bucketName> \s

        Where:
            bucketName - The Amazon S3 bucket name.
        """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        s3LockActions = new S3LockActions();
        bucketName = args[0];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Simple Storage Service (S3)
Object Locking Feature Scenario.");
    }
}
```

```
System.out.println("Press Enter to continue...");
scanner.nextLine();
configurationSetup();
System.out.println(DASHES);

System.out.println(DASHES);
setup();
System.out.println("Setup is complete. Press Enter to continue...");
scanner.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Lets present the user with choices.");
System.out.println("Press Enter to continue...");
scanner.nextLine();
demoActionChoices() ;
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Would you like to clean up the resources? (y/n)");
String delAns = scanner.nextLine().trim();
if (delAns.equalsIgnoreCase("y")) {
    cleanup();
    System.out.println("Clean up is complete.");
}

System.out.println("Press Enter to continue...");
scanner.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Amazon S3 Object Locking Workflow is complete.");
System.out.println(DASHES);
}

// Present the user with the demo action choices.
public static void demoActionChoices() {
    String[] choices = {
        "List all files in buckets.",
        "Attempt to delete a file.",
        "Attempt to delete a file with retention period bypass.",
        "Attempt to overwrite a file.",
        "View the object and bucket retention settings for a file.",
        "View the legal hold settings for a file.",
```

```
        "Finish the workflow."
    };

    int choice = 0;
    while (true) {
        System.out.println(DASHES);
        choice = getChoiceResponse("Explore the S3 locking features by
selecting one of the following choices:", choices);
        System.out.println(DASHES);
        System.out.println("You selected "+choices[choice]);
        switch (choice) {
            case 0 -> {
                s3LockActions.listBucketsAndObjects(bucketNames, true);
            }

            case 1 -> {
                System.out.println("Enter the number of the object to
delete:");

                List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                String[] fileKeysArray = fileKeys.toArray(new String[0]);
                int fileChoice = getChoiceResponse(null, fileKeysArray);
                String objectKey = fileKeys.get(fileChoice);
                String bucketName = allFiles.get(fileChoice).getBucketName();
                String version = allFiles.get(fileChoice).getVersion();
                s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
            }

            case 2 -> {
                System.out.println("Enter the number of the object to
delete:");

                List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
                List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
                String[] fileKeysArray = fileKeys.toArray(new String[0]);
                int fileChoice = getChoiceResponse(null, fileKeysArray);
                String objectKey = fileKeys.get(fileChoice);
                String bucketName = allFiles.get(fileChoice).getBucketName();
                String version = allFiles.get(fileChoice).getVersion();
```

```
        s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
    }

    case 3 -> {
        System.out.println("Enter the number of the object to
overwrite:");

        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();

        // Attempt to overwrite the file.
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
            writer.write("This is a modified text.");

        } catch (IOException e) {
            e.printStackTrace();
        }
        s3LockActions.uploadFile(bucketName, objectKey, objectKey);
    }

    case 4 -> {
        System.out.println("Enter the number of the object to
overwrite:");

        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        s3LockActions.getObjectRetention(bucketName, objectKey);
    }

    case 5 -> {
        System.out.println("Enter the number of the object to
view:");
```

```

        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        s3LockActions.getObjectLegalHold(bucketName, objectKey);
        s3LockActions.getBucketObjectLockConfiguration(bucketName);
    }

    case 6 -> {
        System.out.println("Exiting the workflow...");
        return;
    }

    default -> {
        System.out.println("Invalid choice. Please select again.");
    }
}

}

}

// Clean up the resources from the scenario.
private static void cleanup() {
    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
    for (S3InfoObject fileInfo : allFiles) {
        String bucketName = fileInfo.getBucketName();
        String key = fileInfo.getKeyName();
        String version = fileInfo.getVersion();
        if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
            ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
            if (legalHold != null) {
                String holdStatus = legalHold.status().name();
                System.out.println(holdStatus);
                if (holdStatus.compareTo("ON") == 0) {
                    s3LockActions.modifyObjectLegalHold(bucketName, key,
false);
                }
            }
        }
    }
}

```

```
        // Check for a retention period.
        ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
        boolean hasRetentionPeriod ;
        hasRetentionPeriod = retention != null;
        s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);

    } else {
        System.out.println(bucketName +" objects do not have a legal
lock");
        s3LockActions.deleteObjectFromBucket(bucketName, key,false,
version);
    }
}

// Delete the buckets.
System.out.println("Delete "+bucketName);
for (String bucket : bucketNames){
    s3LockActions.deleteBucketByName(bucket);
}
}

private static void setup() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("""
        For this workflow, we will use the AWS SDK for Java to create
several S3
        buckets and files to demonstrate working with S3 locking
features.
        """);

    System.out.println("S3 buckets can be created either with or without
object lock enabled.");
    System.out.println("Press Enter to continue...");
    scanner.nextLine();

    // Create three S3 buckets.
    s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
    s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
    s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();
}
```

```
        System.out.println("Bucket "+bucketNames.get(2) +" will be configured to
use object locking with a default retention period.");
        s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
        s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
        System.out.println("Press Enter to continue.");
        scanner.nextLine();

        // Upload some files to the buckets.
        System.out.println("Now let's add some test files:");
        String fileName = "exampleFile.txt";
        int fileCount = 2;
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
            writer.write("This is a sample file for uploading to a bucket.");

        } catch (IOException e) {
            e.printStackTrace();
        }

        for (String bucketName : bucketNames){
            for (int i = 0; i < fileCount; i++) {
                // Get the file name without extension.
                String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
                int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
                if (extensionIndex > 0) {
                    fileNameWithoutExtension =
fileNameWithoutExtension.substring(0, extensionIndex);
                }

                // Create the numbered file names.
                String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
                fileNames.add(numberedFileName);
                s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
            }
        }

        String question = null;
```



```

        System.out.print("Press Enter to continue...");
        scanner.nextLine();
        System.out.println("Now we can set some object lock policies on
individual files:");
        for (String bucketName : bucketNames) {
            for (int i = 0; i < fileNames.size(); i++){

                // No modifications to the objects in the first bucket.
                if (!bucketName.equals(bucketNames.get(0))) {
                    String exampleFileName = fileNames.get(i);
                    switch (i) {
                        case 0 -> {
                            question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";
                            System.out.println(question);
                            String ans = scanner.nextLine().trim();
                            if (ans.equalsIgnoreCase("y")) {
                                System.out.println("**** You have selected to put
a legal hold " + exampleFileName);

                                // Set a legal hold.
                                s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
                            }
                        }
                        case 1 -> {
                            """"
                                Would you like to add a 1 day Governance
retention period to %s in %s (y/n)?
                                Reminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.
                                """".formatted(exampleFileName, bucketName);
                            System.out.println(question);
                            String ans2 = scanner.nextLine().trim();
                            if (ans2.equalsIgnoreCase("y")) {

                                s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```
}

// Get file extension.
private static String getFileExtension(String fileName) {
    int dotIndex = fileName.lastIndexOf('.');
    if (dotIndex > 0) {
        return fileName.substring(dotIndex);
    }
    return "";
}

public static void configurationSetup() {
    String noLockBucketName = bucketName + "-no-lock";
    String lockEnabledBucketName = bucketName + "-lock-enabled";
    String retentionAfterCreationBucketName = bucketName + "-retention-after-creation";
    bucketNames.add(noLockBucketName);
    bucketNames.add(lockEnabledBucketName);
    bucketNames.add(retentionAfterCreationBucketName);
}

public static int getChoiceResponse(String question, String[] choices) {
    Scanner scanner = new Scanner(System.in);
    if (question != null) {
        System.out.println(question);
        for (int i = 0; i < choices.length; i++) {
            System.out.println("\t" + (i + 1) + ". " + choices[i]);
        }
    }

    int choiceNumber = 0;
    while (choiceNumber < 1 || choiceNumber > choices.length) {
        String choice = scanner.nextLine();
        try {
            choiceNumber = Integer.parseInt(choice);
        } catch (NumberFormatException e) {
            System.out.println("Invalid choice. Please enter a valid number.");
        }
    }

    return choiceNumber - 1;
}
}
```

Una classe wrapper per le funzioni S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
import
    software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import
    software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import
    software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
```

```
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
public class S3LockActions {

    private static S3Client getClient() {
        return S3Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }

    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
    {
        // Calculate the instant one day from now.
        Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

        // Convert the Instant to a ZonedDateTime object with a specific time
        zone.
        ZonedDateTime zonedDateTime =
        futureInstant.atZone(ZoneId.systemDefault());

        // Define a formatter for human-readable output.
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
        HH:mm:ss");

        // Format the ZonedDateTime object to a human-readable date string.
        String humanReadableDate = formatter.format(zonedDateTime);

        // Print the formatted date string.
        System.out.println("Formatted Date: " + humanReadableDate);
        ObjectLockRetention retention = ObjectLockRetention.builder()
            .mode(ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(futureInstant)
            .build();
    }
}
```

```
        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .retention(retention)
            .build();

        getClient().putObjectRetention(retentionRequest);
        System.out.println("Set retention for "+objectKey+" in "+bucketName+"
until "+humanReadableDate+".");
    }

    // Get the legal hold details for an S3 object.
    public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
        try {
            GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
            System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
                ":\n\tStatus: " + response.legalHold().status());
            return response.legalHold();

        } catch (S3Exception ex) {
            System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
        }

        return null;
    }

    // Create a new Amazon S3 bucket with object lock options.
    public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
        S3Waiter s3Waiter = getClient().waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectLockEnabledForBucket(enableObjectLock)
```

```

        .build();

        getClient().createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        s3Waiter.waitUntilBucketExists(bucketRequestWait);
        System.out.println(bucketName + " is ready");
    }

    public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
        Boolean interactive) {
        AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
        return bucketNames.stream()
            .flatMap(bucketName ->
                listBucketObjectsAndVersions(bucketName).versions().stream()
                    .map(version -> {
                        S3InfoObject s3InfoObject = new S3InfoObject();
                        s3InfoObject.setBucketName(bucketName);
                        s3InfoObject.setVersion(version.versionId());
                        s3InfoObject.setKeyName(version.key());
                        return s3InfoObject;
                    })
            )
            .peek(s3InfoObject -> {
                int i = counter.incrementAndGet(); // Increment and get the
updated value.
                if (interactive) {
                    System.out.println(i + ": " + s3InfoObject.getKeyName());
                    System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
                    System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
                }
            })
            .collect(Collectors.toList());
    }

    public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
        ListObjectVersionsRequest versionsRequest =
        ListObjectVersionsRequest.builder()
            .bucket(bucketName)

```

```
        .build();

    return getClient().listObjectVersions(versionsRequest);
}

// Set or modify a retention period on an S3 bucket.
public void modifyBucketDefaultRetention(String bucketName) {
    VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
        .mfaDelete(MFADelete.DISABLED)
        .status(BucketVersioningStatus.ENABLED)
        .build();

    PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

    getClient().putBucketVersioning(versioningRequest);
    DefaultRetention rention = DefaultRetention.builder()
        .days(1)
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .build();

    ObjectLockRule lockRule = ObjectLockRule.builder()
        .defaultRetention(rention)
        .build();

    ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
        .objectLockEnabled(ObjectLockEnabled.ENABLED)
        .rule(lockRule)
        .build();

    PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .objectLockConfiguration(objectLockConfiguration)
        .build();

    getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
}
```

```
        System.out.println("Added a default retention to bucket "+bucketName
+ ".");
    }

    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
                    .build())
                .build();

            getClient().putObjectLockConfiguration(request);
            System.out.println("Successfully enabled object lock on
"+bucketName);

        } catch (S3Exception ex) {
            System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
        }
    }

    public void uploadFile(String bucketName, String objectName, String filePath)
    {
        Path file = Paths.get(filePath);
        PutObjectRequest request = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectName)
```



```
        .checksumAlgorithm(ChecksumAlgorithm.SHA256)
        .build();

    PutObjectResponse response = getClient().putObject(request, file);
    if (response != null) {
        System.out.println("\tSuccessfully uploaded " + objectName + " to " +
            bucketName + ".");
    } else {
        System.out.println("\tCould not upload " + objectName + " to " +
            bucketName + ".");
    }
}

// Set or modify a legal hold on an object in an S3 bucket.
public void modifyObjectLegalHold(String bucketName, String objectKey,
    boolean legalHoldOn) {
    ObjectLockLegalHold legalHold ;
    if (legalHoldOn) {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.ON)
            .build();
    } else {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.OFF)
            .build();
    }

    PutObjectLegalHoldRequest legalHoldRequest =
        PutObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .legalHold(legalHold)
            .build();

    getClient().putObjectLegalHold(legalHoldRequest) ;
    System.out.println("Modified legal hold for " + objectKey + " in
        "+bucketName + ".");
}

// Delete an object from a specific bucket.
public void deleteObjectFromBucket(String bucketName, String objectKey,
    boolean hasRetention, String versionId) {
    try {
        DeleteObjectRequest objectRequest;
```

```
        if (hasRetention) {
            objectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .versionId(versionId)
                .bypassGovernanceRetention(true)
                .build();
        } else {
            objectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .versionId(versionId)
                .build();
        }

        getClient().deleteObject(objectRequest) ;
        System.out.println("The object was successfully deleted");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

// Get the retention period for an S3 object.
public ObjectLockRetention getObjectRetention(String bucketName, String key){
    try {
        GetObjectRetentionRequest retentionRequest =
        GetObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        GetObjectRetentionResponse response =
        getClient().getObjectRetention(retentionRequest);
        System.out.println("tObject retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}
```

```
public void deleteBucketByName(String bucketName) {
    try {
        DeleteBucketRequest request = DeleteBucketRequest.builder()
            .bucket(bucketName)
            .build();

        getClient().deleteBucket(request);
        System.out.println(bucketName + " was deleted.");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}

// Get the object lock configuration details for an S3 bucket.
public void getBucketObjectLockConfiguration(String bucketName) {
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .build();

    GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
    System.out.println("Bucket object lock config for "+bucketName+": ");
    System.out.println("\tEnabled:
"+response.getObjectLockConfiguration().getObjectLockEnabled());
    System.out.println("\tRule: "+
response.getObjectLockConfiguration().rule().getDefaultRetention());
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK for Java 2.x .
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Punto di ingresso per lo scenario (index.js). In questo modo vengono orchestrate tutte le fasi. Visita GitHub per vedere i dettagli di implementazione di Scenario ScenarioInput, ScenarioOutput, e ScenarioAction.

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "../welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  confirmSetLegalHoldFileEnabled,
  confirmSetLegalHoldFileRetention,
  confirmSetRetentionPeriodFileEnabled,
  confirmSetRetentionPeriodFileRetention,
  confirmUpdateLockPolicy,
  confirmUpdateRetention,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
  setLegalHoldFileEnabledAction,
  setLegalHoldFileRetentionAction,
  setRetentionPeriodFileEnabledAction,
  setRetentionPeriodFileRetentionAction,
  updateLockPolicy,
  updateLockPolicyAction,
```

```
    updateRetention,
    updateRetentionAction,
  } from "./setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new scenarios.Scenario(
      "S3 Object Locking - Deploy",
      [
        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        getBucketPrefix(scenarios),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        updateRetention(scenarios),
        confirmUpdateRetention(scenarios),
        exitOnFalse(scenarios, "confirmUpdateRetention"),
        updateRetentionAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        updateLockPolicy(scenarios),
        confirmUpdateLockPolicy(scenarios),
        exitOnFalse(scenarios, "confirmUpdateLockPolicy"),
        updateLockPolicyAction(scenarios, client),
        confirmSetLegalHoldFileEnabled(scenarios),
        setLegalHoldFileEnabledAction(scenarios, client),
        confirmSetRetentionPeriodFileEnabled(scenarios),
        setRetentionPeriodFileEnabledAction(scenarios, client),
        confirmSetLegalHoldFileRetention(scenarios),
        setLegalHoldFileRetentionAction(scenarios, client),
        confirmSetRetentionPeriodFileRetention(scenarios),
        setRetentionPeriodFileRetentionAction(scenarios, client),
        saveState,
      ]
    ),
  };
}
```

```

    ],
    initialState,
  ),
  demo: new scenarios.Scenario(
    "S3 Object Locking - Demo",
    [loadState, replAction(scenarios, client)],
    initialState,
  ),
  clean: new scenarios.Scenario(
    "S3 Object Locking - Destroy",
    [
      loadState,
      confirmCleanup(scenarios),
      exitOnFalse(scenarios, "confirmCleanup"),
      cleanupAction(scenarios, client),
    ],
    initialState,
  ),
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const objectLockingScenarios = getWorkflowStages(Scenarios);
  Scenarios.parseScenarioArgs(objectLockingScenarios, {
    name: "Amazon S3 object locking workflow",
    description:
      "Work with Amazon Simple Storage Service (Amazon S3) object locking features.",
    synopsis:
      "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-v|--verbose]",
  });
}

```

Restituisce i messaggi di benvenuto alla console (welcome.steps.js).

```
/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    "Welcome to the Amazon Simple Storage Service (S3) Object Locking Feature
    Scenario. For this workflow, we will use the AWS SDK for JavaScript to create
    several S3 buckets and files to demonstrate working with S3 locking features.",
    { header: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };
```

Distribuisce bucket, oggetti e impostazioni dei file (setup.steps.js).

```
import {
  BucketVersioningStatus,
  ChecksumAlgorithm,
  CreateBucketCommand,
  MFADeleteStatus,
  PutBucketVersioningCommand,
  PutObjectCommand,
  PutObjectLockConfigurationCommand,
  PutObjectLegalHoldCommand,
  PutObjectRetentionCommand,
  ObjectLockLegalHoldStatus,
  ObjectLockRetentionMode,
```

```

    GetBucketVersioningCommand,
    BucketAlreadyExists,
    BucketAlreadyOwnedByYou,
    S3ServiceException,
    waitUntilBucketExists,
} from "@aws-sdk/client-s3";

import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const getBucketPrefix = (scenarios) =>
  new scenarios.ScenarioInput(
    "bucketPrefix",
    "Provide a prefix that will be used for bucket creation.",
    { type: "input", default: "amzn-s3-demo-bucket" },
  );

/**
 * @param {Scenarios} scenarios
 */
const createBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "createBuckets",
    (state) => `The following buckets will be created:
      ${state.bucketPrefix}-no-lock with object lock False.
      ${state.bucketPrefix}-lock-enabled with object lock True.
      ${state.bucketPrefix}-retention-after-creation with object lock False.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmCreateBuckets = (scenarios) =>

```



```
new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
  type: "confirm",
});

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {
    const noLockBucketName = `${state.bucketPrefix}-no-lock`;
    const lockEnabledBucketName = `${state.bucketPrefix}-lock-enabled`;
    const retentionBucketName = `${state.bucketPrefix}-retention-after-creation`;

    try {
      await client.send(new CreateBucketCommand({ Bucket: noLockBucketName }));
      await waitUntilBucketExists({ client }, { Bucket: noLockBucketName });
      await client.send(
        new CreateBucketCommand({
          Bucket: lockEnabledBucketName,
          ObjectLockEnabledForBucket: true,
        }),
      );
      await waitUntilBucketExists(
        { client },
        { Bucket: lockEnabledBucketName },
      );
      await client.send(
        new CreateBucketCommand({ Bucket: retentionBucketName }),
      );
      await waitUntilBucketExists({ client }, { Bucket: retentionBucketName });

      state.noLockBucketName = noLockBucketName;
      state.lockEnabledBucketName = lockEnabledBucketName;
      state.retentionBucketName = retentionBucketName;
    } catch (caught) {
      if (
        caught instanceof BucketAlreadyExists ||
        caught instanceof BucketAlreadyOwnedByYou
      ) {
        console.error(`${caught.name}: ${caught.message}`);
        state.earlyExit = true;
      } else {
        throw caught;
      }
    }
  })
```

```

    }
  }
});

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",
    (state) => `The following test files will be created:
      file0.txt in ${state.bucketPrefix}-no-lock.
      file1.txt in ${state.bucketPrefix}-no-lock.
      file0.txt in ${state.bucketPrefix}-lock-enabled.
      file1.txt in ${state.bucketPrefix}-lock-enabled.
      file0.txt in ${state.bucketPrefix}-retention-after-creation.
      file1.txt in ${state.bucketPrefix}-retention-after-creation.` ,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmPopulateBuckets",
    "Populate the buckets?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
    try {
      await client.send(
        new PutObjectCommand({
          Bucket: state.noLockBucketName,
          Key: "file0.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
    }
  });

```

```
);
await client.send(
    new PutObjectCommand({
        Bucket: state.noLockBucketName,
        Key: "file1.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
    }),
);
await client.send(
    new PutObjectCommand({
        Bucket: state.lockEnabledBucketName,
        Key: "file0.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
    }),
);
await client.send(
    new PutObjectCommand({
        Bucket: state.lockEnabledBucketName,
        Key: "file1.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
    }),
);
await client.send(
    new PutObjectCommand({
        Bucket: state.retentionBucketName,
        Key: "file0.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
    }),
);
await client.send(
    new PutObjectCommand({
        Bucket: state.retentionBucketName,
        Key: "file1.txt",
        Body: "Content",
        ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
    }),
);
} catch (caught) {
    if (caught instanceof S3ServiceException) {
        console.error(
```

```

        `Error from S3 while uploading object.  ${caught.name}:
        ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
});

/**
 * @param {Scenarios} scenarios
 */
const updateRetention = (scenarios) =>
    new scenarios.ScenarioOutput(
        "updateRetention",
        (state) => `A bucket can be configured to use object locking with a default
        retention period.
        A default retention period will be configured for ${state.bucketPrefix}-
        retention-after-creation.` ,
        { preformatted: true },
    );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateRetention = (scenarios) =>
    new scenarios.ScenarioInput(
        "confirmUpdateRetention",
        "Configure default retention period?",
        { type: "confirm" },
    );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateRetentionAction = (scenarios, client) =>
    new scenarios.ScenarioAction("updateRetentionAction", async (state) => {
        await client.send(
            new PutBucketVersioningCommand({
                Bucket: state.retentionBucketName,
                VersioningConfiguration: {
                    MFADelete: MFADeleteStatus.Disabled,
                    Status: BucketVersioningStatus.Enabled,

```

```

    },
  )),
);

const getBucketVersioning = new GetBucketVersioningCommand({
  Bucket: state.retentionBucketName,
});

await retry({ intervalInMs: 500, maxRetries: 10 }, async () => {
  const { Status } = await client.send(getBucketVersioning);
  if (Status !== "Enabled") {
    throw new Error("Bucket versioning is not enabled.");
  }
});

await client.send(
  new PutObjectLockConfigurationCommand({
    Bucket: state.retentionBucketName,
    ObjectLockConfiguration: {
      ObjectLockEnabled: "Enabled",
      Rule: {
        DefaultRetention: {
          Mode: "GOVERNANCE",
          Years: 1,
        },
      },
    },
  }),
);
});

/**
 * @param {Scenarios} scenarios
 */
const updateLockPolicy = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateLockPolicy",
    (state) => `Object lock policies can also be added to existing buckets.
An object lock policy will be added to ${state.bucketPrefix}-lock-enabled.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios

```

```

    */
    const confirmUpdateLockPolicy = (scenarios) =>
      new scenarios.ScenarioInput(
        "confirmUpdateLockPolicy",
        "Add object lock policy?",
        { type: "confirm" },
      );

    /**
     * @param {Scenarios} scenarios
     * @param {S3Client} client
     */
    const updateLockPolicyAction = (scenarios, client) =>
      new scenarios.ScenarioAction("updateLockPolicyAction", async (state) => {
        await client.send(
          new PutObjectLockConfigurationCommand({
            Bucket: state.lockEnabledBucketName,
            ObjectLockConfiguration: {
              ObjectLockEnabled: "Enabled",
            },
          }),
        );
      });

    /**
     * @param {Scenarios} scenarios
     * @param {S3Client} client
     */
    const confirmSetLegalHoldFileEnabled = (scenarios) =>
      new scenarios.ScenarioInput(
        "confirmSetLegalHoldFileEnabled",
        (state) =>
          `Would you like to add a legal hold to file0.txt in
          ${state.lockEnabledBucketName}?`,
        {
          type: "confirm",
        },
      );

    /**
     * @param {Scenarios} scenarios
     * @param {S3Client} client
     */
    const setLegalHoldFileEnabledAction = (scenarios, client) =>

```

```

    new scenarios.ScenarioAction(
      "setLegalHoldFileEnabledAction",
      async (state) => {
        await client.send(
          new PutObjectLegalHoldCommand({
            Bucket: state.lockEnabledBucketName,
            Key: "file0.txt",
            LegalHold: {
              Status: ObjectLockLegalHoldStatus.ON,
            },
          }),
        );
        console.log(
          `Modified legal hold for file0.txt in ${state.lockEnabledBucketName}.`,
        );
      },
      { skipWhen: (state) => !state.confirmSetLegalHoldFileEnabled },
    );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetRetentionPeriodFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileEnabled",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
      ${state.lockEnabledBucketName}?
      Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
      able to delete this file or its bucket until the retention period has expired.`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileEnabledAction",
    async (state) => {

```

```

        const retentionDate = new Date();
        retentionDate.setDate(retentionDate.getDate() + 1);
        await client.send(
            new PutObjectRetentionCommand({
                Bucket: state.lockEnabledBucketName,
                Key: "file1.txt",
                Retention: {
                    Mode: ObjectLockRetentionMode.GOVERNANCE,
                    RetainUntilDate: retentionDate,
                },
            }),
        );
        console.log(
            `Set retention for file1.txt in ${state.lockEnabledBucketName} until
            ${retentionDate.toISOString().split("T")[0]}.`,
        );
    },
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileEnabled },
);

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileRetention = (scenarios) =>
    new scenarios.ScenarioInput(
        "confirmSetLegalHoldFileRetention",
        (state) =>
            `Would you like to add a legal hold to file0.txt in
            ${state.retentionBucketName}?`,
        {
            type: "confirm",
        },
    );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileRetentionAction = (scenarios, client) =>
    new scenarios.ScenarioAction(
        "setLegalHoldFileRetentionAction",
        async (state) => {
            await client.send(

```



```

        new PutObjectLegalHoldCommand({
            Bucket: state.retentionBucketName,
            Key: "file0.txt",
            LegalHold: {
                Status: ObjectLockLegalHoldStatus.ON,
            },
        }),
    );
    console.log(
        `Modified legal hold for file0.txt in ${state.retentionBucketName}.`,
    );
},
{ skipWhen: (state) => !state.confirmSetLegalHoldFileRetention },
);

/**
 * @param {Scenarios} scenarios
 */
const confirmSetRetentionPeriodFileRetention = (scenarios) =>
    new scenarios.ScenarioInput(
        "confirmSetRetentionPeriodFileRetention",
        (state) =>
            `Would you like to add a 1 day Governance retention period to file1.txt in
            ${state.retentionBucketName}?
            Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
            able to delete this file or its bucket until the retention period has expired.`,
        {
            type: "confirm",
        },
    );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileRetentionAction = (scenarios, client) =>
    new scenarios.ScenarioAction(
        "setRetentionPeriodFileRetentionAction",
        async (state) => {
            const retentionDate = new Date();
            retentionDate.setDate(retentionDate.getDate() + 1);
            await client.send(
                new PutObjectRetentionCommand({
                    Bucket: state.retentionBucketName,

```

```
        Key: "file1.txt",
        Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
        },
        BypassGovernanceRetention: true,
    )),
);
console.log(
    `Set retention for file1.txt in ${state.retentionBucketName} until
    ${retentionDate.toISOString().split("T")[0]}.`,
);
},
{ skipWhen: (state) => !state.confirmSetRetentionPeriodFileRetention },
);

export {
    getBucketPrefix,
    createBuckets,
    confirmCreateBuckets,
    createBucketsAction,
    populateBuckets,
    confirmPopulateBuckets,
    populateBucketsAction,
    updateRetention,
    confirmUpdateRetention,
    updateRetentionAction,
    updateLockPolicy,
    confirmUpdateLockPolicy,
    updateLockPolicyAction,
    confirmSetLegalHoldFileEnabled,
    setLegalHoldFileEnabledAction,
    confirmSetRetentionPeriodFileEnabled,
    setRetentionPeriodFileEnabledAction,
    confirmSetLegalHoldFileRetention,
    setLegalHoldFileRetentionAction,
    confirmSetRetentionPeriodFileRetention,
    setRetentionPeriodFileRetentionAction,
};
```

Visualizza ed elimina i file nei bucket (repl.steps.js).

```
import {
  ChecksumAlgorithm,
  DeleteObjectCommand,
  GetObjectLegalHoldCommand,
  GetObjectLockConfigurationCommand,
  GetObjectRetentionCommand,
  ListObjectVersionsCommand,
  PutObjectCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  DELETE_FILE: 2,
  DELETE_FILE_WITH_RETENTION: 3,
  OVERWRITE_FILE: 4,
  VIEW_RETENTION_SETTINGS: 5,
  VIEW_LEGAL_HOLD_SETTINGS: 6,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new scenarios.ScenarioInput(
    "replChoice",
    "Explore the S3 locking features by selecting one of the following choices",
    {
      type: "select",
      choices: [
        { name: "List all files in buckets", value: choices.LIST_ALL_FILES },
        { name: "Attempt to delete a file.", value: choices.DELETE_FILE },
        {
          name: "Attempt to delete a file with retention period bypass.",
          value: choices.DELETE_FILE_WITH_RETENTION,
        }
      ]
    }
  )
```

```

    },
    { name: "Attempt to overwrite a file.", value: choices.OVERWRITE_FILE },
    {
      name: "View the object and bucket retention settings for a file.",
      value: choices.VIEW_RETENTION_SETTINGS,
    },
    {
      name: "View the legal hold settings for a file.",
      value: choices.VIEW_LEGAL_HOLD_SETTINGS,
    },
    { name: "Finish the workflow.", value: choices.EXIT },
  ],
},
);

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
  /** @type {{bucket: string, key: string, version: string}[]} */
  const files = [];
  for (const bucket of buckets) {
    const objectsResponse = await client.send(
      new ListObjectVersionsCommand({ Bucket: bucket }),
    );
    for (const version of objectsResponse.Versions || []) {
      const { Key, VersionId } = version;
      files.push({ bucket, key: Key, version: VersionId });
    }
  }

  return files;
};

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const replAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [

```

```
state.noLockBucketName,
state.lockEnabledBucketName,
state.retentionBucketName,
]);

const fileInput = new scenarios.ScenarioInput(
  "selectedFile",
  "Select a file:",
  {
    type: "select",
    choices: files.map((file, index) => ({
      name: `${index + 1}: ${file.bucket}: ${file.key} (version: ${
        file.version
      })`,
      value: index,
    })),
  },
);

const { replChoice } = state;

switch (replChoice) {
  case choices.LIST_ALL_FILES: {
    const files = await getAllFiles(client, [
      state.noLockBucketName,
      state.lockEnabledBucketName,
      state.retentionBucketName,
    ]);
    state.replOutput = files
      .map(
        (file) =>
          `${file.bucket}: ${file.key} (version: ${file.version})`,
      )
      .join("\n");
    break;
  }
  case choices.DELETE_FILE: {
    /** @type {number} */
    const fileToDelete = await fileInput.handle(state);
    const selectedFile = files[fileToDelete];
    try {
      await client.send(
        new DeleteObjectCommand({
          Bucket: selectedFile.bucket,
```

```

        Key: selectedFile.key,
        VersionId: selectedFile.version,
    )),
    );
    state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
    } catch (err) {
        state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
    }
    break;
}
case choices.DELETE_FILE_WITH_RETENTION: {
    /** @type {number} */
    const fileToDelete = await fileInput.handle(state);
    const selectedFile = files[fileToDelete];
    try {
        await client.send(
            new DeleteObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
                BypassGovernanceRetention: true,
            )),
        );
        state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
    } catch (err) {
        state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
    }
    break;
}
case choices.OVERWRITE_FILE: {
    /** @type {number} */
    const fileToOverwrite = await fileInput.handle(state);
    const selectedFile = files[fileToOverwrite];
    try {
        await client.send(
            new PutObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                Body: "New content",
                ChecksumAlgorithm: ChecksumAlgorithm.SHA256,

```

```

        })),
    );
    state.replOutput = `Overwrote ${selectedFile.key} in
${selectedFile.bucket}.`;
    } catch (err) {
        state.replOutput = `Unable to overwrite object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
    }
    break;
}
case choices.VIEW_RETENTION_SETTINGS: {
    /** @type {number} */
    const fileToView = await fileInput.handle(state);
    const selectedFile = files[fileToView];
    try {
        const retention = await client.send(
            new GetObjectRetentionCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
            })),
    );
    const bucketConfig = await client.send(
        new GetObjectLockConfigurationCommand({
            Bucket: selectedFile.bucket,
        })),
    );
    state.replOutput = `Object retention for ${selectedFile.key}
in ${selectedFile.bucket}: ${retention.Retention?.Mode} until
${retention.Retention?.RetainUntilDate?.toISOString()}.
Bucket object lock config for ${selectedFile.bucket} in ${selectedFile.bucket}:
Enabled: ${bucketConfig.ObjectLockConfiguration?.ObjectLockEnabled}
Rule:
${JSON.stringify(bucketConfig.ObjectLockConfiguration?.Rule?.DefaultRetention)}`;
    } catch (err) {
        state.replOutput = `Unable to fetch object lock retention:
'${err.message}'`;
    }
    break;
}
case choices.VIEW_LEGAL_HOLD_SETTINGS: {
    /** @type {number} */
    const fileToView = await fileInput.handle(state);
    const selectedFile = files[fileToView];

```

```

    try {
      const legalHold = await client.send(
        new GetObjectLegalHoldCommand({
          Bucket: selectedFile.bucket,
          Key: selectedFile.key,
          VersionId: selectedFile.version,
        }),
      );
      state.replOutput = `Object legal hold for ${selectedFile.key} in
${selectedFile.bucket}: Status: ${legalHold.LegalHold?.Status}`;
    } catch (err) {
      state.replOutput = `Unable to fetch legal hold: '${err.message}'`;
    }
    break;
  }
  default:
    throw new Error(`Invalid replChoice: ${replChoice}`);
}
},
{
  whileConfig: {
    whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
    input: replInput(scenarios),
    output: new scenarios.ScenarioOutput(
      "REPL output",
      (state) => state.replOutput,
      { preformatted: true },
    ),
  },
},
);

export { replInput, replAction, choices };

```

Elimina tutte le risorse create (clean.steps.js).

```

import {
  DeleteObjectCommand,
  DeleteBucketCommand,
  ListObjectVersionsCommand,
  GetObjectLegalHoldCommand,
  GetObjectRetentionCommand,

```



```
    PutObjectLegalHoldCommand,
  } from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { noLockBucketName, lockEnabledBucketName, retentionBucketName } =
      state;

    const buckets = [
      noLockBucketName,
      lockEnabledBucketName,
      retentionBucketName,
    ];

    for (const bucket of buckets) {
      /** @type {import("@aws-sdk/client-s3").ListObjectVersionsCommandOutput} */
      let objectsResponse;

      try {
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
      }
    }
  });
```

```
    } catch (e) {
      if (e instanceof Error && e.name === "NoSuchBucket") {
        console.log("Object's bucket has already been deleted.");
        continue;
      }
      throw e;
    }

    for (const version of objectsResponse.Versions || []) {
      const { Key, VersionId } = version;

      try {
        const legalHold = await client.send(
          new GetObjectLegalHoldCommand({
            Bucket: bucket,
            Key,
            VersionId,
          }),
        );

        if (legalHold.LegalHold?.Status === "ON") {
          await client.send(
            new PutObjectLegalHoldCommand({
              Bucket: bucket,
              Key,
              VersionId,
              LegalHold: {
                Status: "OFF",
              },
            }),
          );
        }
      } catch (err) {
        console.log(
          `Unable to fetch legal hold for ${Key} in ${bucket}:` +
            `${err.message}`,
        );
      }

      try {
        const retention = await client.send(
          new GetObjectRetentionCommand({
            Bucket: bucket,
            Key,
```

```
        VersionId,
      )),
    );

    if (retention.Retention?.Mode === "GOVERNANCE") {
      await client.send(
        new DeleteObjectCommand({
          Bucket: bucket,
          Key,
          VersionId,
          BypassGovernanceRetention: true,
        )),
      );
    }
  } catch (err) {
    console.log(
      `Unable to fetch object lock retention for ${Key} in ${bucket}:
      '${err.message}'`,
    );
  }

  await client.send(
    new DeleteObjectCommand({
      Bucket: bucket,
      Key,
      VersionId,
    )),
  );
}

await client.send(new DeleteBucketCommand({ Bucket: bucket }));
console.log(`Delete for ${bucket} complete.`);
}
});

export { confirmCleanup, cleanupAction };
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per JavaScript .
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)

- [GetObjectRetention](#)
- [PutObjectLegalHold](#)
- [PutObjectLockConfiguration](#)
- [PutObjectRetention](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Effettua richieste condizionali per Amazon S3 utilizzando un AWS SDK

Gli esempi di codice seguenti mostrano come aggiungere precondizioni alle richieste Amazon S3.

.NET

SDK per .NET

Note

C'è altro su GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegue uno scenario interattivo che dimostra le funzionalità delle richieste condizionali di Amazon S3.

```
using Amazon.S3;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ConditionalRequestsScenario;

public static class S3ConditionalRequestsScenario
{
    /*
```

Before running this .NET code example, set up your development environment, including your credentials.

This example demonstrates the use of conditional requests for S3 operations. You can use conditional requests to add preconditions to S3 read requests to return or copy

an object based on its Entity tag (ETag), or last modified date.

You can use a conditional write requests to prevent overwrites by ensuring there is no existing object with the same key.

*/

```
public static S3ActionsWrapper _s3ActionsWrapper = null!;  
public static IConfiguration _configuration = null!;  
public static string _resourcePrefix = null!;  
public static string _sourceBucketName = null!;  
public static string _destinationBucketName = null!;  
public static string _sampleObjectKey = null!;  
public static string _sampleObjectEtag = null!;  
public static bool _interactive = true;  
  
public static async Task Main(string[] args)  
{  
    // Set up dependency injection for the Amazon service.  
    using var host = Host.CreateDefaultBuilder(args)  
        .ConfigureLogging(logging =>  
            logging.AddFilter("System", LogLevel.Debug)  
                .AddFilter<DebugLoggerProvider>("Microsoft",  
LogLevel.Information)  
                .AddFilter<ConsoleLoggerProvider>("Microsoft",  
LogLevel.Trace))  
        .ConfigureServices((_, services) =>  
            services.AddAWSService<IAmazonS3>()  
                .AddTransient<S3ActionsWrapper>()  
        )  
        .Build();  
  
    _configuration = new ConfigurationBuilder()  
        .SetBasePath(Directory.GetCurrentDirectory())  
        .AddJsonFile("settings.json") // Load settings from .json file.  
        .AddJsonFile("settings.local.json",  
            true) // Optionally, load local settings.  
        .Build();
```

```

        ServicesSetup(host);

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)
Conditional Requests Feature Scenario.");
            Console.WriteLine(new string('-', 80));
            ConfigurationSetup();
            _sampleObjectEtag = await Setup(_sourceBucketName,
            _destinationBucketName, _sampleObjectKey);

            await DisplayDemoChoices(_sourceBucketName, _destinationBucketName,
            _sampleObjectKey, _sampleObjectEtag, 0);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Cleaning up resources.");
            Console.WriteLine(new string('-', 80));
            await Cleanup(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Amazon S3 Conditional Requests Feature Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem: {ex.Message}");
            await CleanupScenario(_sourceBucketName, _destinationBucketName);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
    }

    /// <summary>

```

```

    /// Any setup operations needed.
    /// </summary>
    public static void ConfigurationSetup()
    {
        _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

        _sourceBucketName = _resourcePrefix + "-source";
        _destinationBucketName = _resourcePrefix + "-dest";
        _sampleObjectKey = _resourcePrefix + "-sample-object.txt";
    }

    /// <summary>
    /// Sets up the scenario by creating a source and destination bucket, and
    uploading a test file to the source bucket.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    /// <param name="objectKey">The name of the test file to add to the source
    bucket.</param>
    /// <returns>The ETag of the uploaded test file.</returns>
    public static async Task<string> Setup(string sourceBucket, string
    destBucket, string objectKey)
    {
        Console.WriteLine(
            "\nFor this scenario, we will use the AWS SDK for .NET to create
            several S3\n" +
            "buckets and files to demonstrate working with S3 conditional
            requests.\n" +
            "This example demonstrates the use of conditional requests for S3
            operations.\r\n" +
            "You can use conditional requests to add preconditions to S3 read
            requests to return or copy\r\n" +
            "an object based on its Entity tag (ETag), or last modified date. \r
            \n" +
            "You can use a conditional write requests to prevent overwrites by
            ensuring \r\n" +
            "there is no existing object with the same key. \r\n\r\n" +
            "This example will allow you to perform conditional reads\r\n" +
            "and writes that will succeed or fail based on your selected options.
            \r\n\r\n" +
            "Sample buckets and a sample object will be created as part of the
            example.");

        Console.WriteLine(new string('-', 80));
    }

```

```
        Console.WriteLine("Press Enter when you are ready to start.");
        if (_interactive)
            Console.ReadLine();

        await _s3ActionsWrapper.CreateBucketWithName(sourceBucket);
        await _s3ActionsWrapper.CreateBucketWithName(destBucket);

        var eTag = await _s3ActionsWrapper.PutObjectConditional(objectKey,
sourceBucket,
            "Test file content.");

        return eTag;
    }

    /// <summary>
    /// Cleans up the scenario by deleting the source and destination buckets.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    public static async Task CleanupScenario(string sourceBucket, string
destBucket)
    {
        await _s3ActionsWrapper.CleanupBucketByName(sourceBucket);
        await _s3ActionsWrapper.CleanupBucketByName(destBucket);
    }

    /// <summary>
    /// Displays a list of the objects in the test buckets.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    public static async Task DisplayBuckets(string sourceBucket, string
destBucket)
    {
        await _s3ActionsWrapper.ListBucketContentsByName(sourceBucket);
        await _s3ActionsWrapper.ListBucketContentsByName(destBucket);
    }

    /// <summary>
    /// Displays the menu of conditional request options for the user.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
```



```

    /// <param name="objectKey">The key of the test object in the source
    bucket.</param>
    /// <param name="etag">The ETag of the test object in the source bucket.</
    param>
    public static async Task DisplayDemoChoices(string sourceBucket, string
    destBucket, string objectKey, string etag, int defaultChoice)
    {
        var actions = new[]
        {
            "Print a list of bucket items.",
            "Perform a conditional read.",
            "Perform a conditional copy.",
            "Perform a conditional write.",
            "Clean up and exit."
        };

        var conditions = new[]
        {
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should
fail.",
            "If-Modified-Since: using yesterday's date. This condition should
succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
fail."
        };

        var conditionTypes = new[]
        {
            S3ConditionType.IfMatch,
            S3ConditionType.IfNoneMatch,
            S3ConditionType.IfModifiedSince,
            S3ConditionType.IfUnmodifiedSince,
        };

        var yesterdayDate = DateTime.UtcNow.AddDays(-1);

        int choice;
        while ((choice = GetChoiceResponse("\nExplore the S3 conditional request
features by selecting one of the following choices:", actions, defaultChoice)) != 4)
        {
            switch (choice)
            {

```

```
        case 0:
            Console.WriteLine("Listing the objects and buckets.");
            await DisplayBuckets(sourceBucket, destBucket);
            break;
        case 1:
            int conditionTypeIndex = GetChoiceResponse("Perform a
conditional read:", conditions, 1);
            if (conditionTypeIndex == 0 || conditionTypeIndex == 1)
            {
                await _s3ActionsWrapper.GetObjectConditional(objectKey,
sourceBucket, conditionTypes[conditionTypeIndex], null, _sampleObjectEtag);
            }
            else if (conditionTypeIndex == 2 || conditionTypeIndex == 3)
            {
                await _s3ActionsWrapper.GetObjectConditional(objectKey,
sourceBucket, conditionTypes[conditionTypeIndex], yesterdayDate);
            }
            break;
        case 2:
            int copyConditionTypeIndex = GetChoiceResponse("Perform a
conditional copy:", conditions, 1);
            string destKey = GetStringResponse("Enter an object key:",
"sampleObjectKey");
            if (copyConditionTypeIndex == 0 || copyConditionTypeIndex ==
1)
            {
                await _s3ActionsWrapper.CopyObjectConditional(objectKey,
destKey, sourceBucket, destBucket, conditionTypes[copyConditionTypeIndex], null,
etag);
            }
            else if (copyConditionTypeIndex == 2 ||
copyConditionTypeIndex == 3)
            {
                await _s3ActionsWrapper.CopyObjectConditional(objectKey,
destKey, sourceBucket, destBucket, conditionTypes[copyConditionTypeIndex],
yesterdayDate);
            }
            break;
        case 3:
            Console.WriteLine("Perform a conditional write using
IfNoneMatch condition on the object key.");
            Console.WriteLine("If the key is a duplicate, the write will
fail.");
```

```

        string newObjectKey = GetStringResponse("Enter an object
key:", "newObjectKey");
        await _s3ActionsWrapper.PutObjectConditional(newObjectKey,
sourceBucket, "Conditional write example data.");
        break;
    }

    if (!_interactive)
    {
        break;
    }
}

Console.WriteLine("Proceeding to cleanup.");
}

// <summary>
/// Clean up the resources from the scenario.
/// </summary>
/// <param name="interactive">True to run as interactive.</param>
/// <returns>True if successful.</returns>
public static async Task<bool> Cleanup(bool interactive)
{
    Console.WriteLine(new string('-', 80));

    if (!interactive || GetYesNoResponse("Do you want to clean up all files
and buckets? (y/n) "))
    {
        await _s3ActionsWrapper.CleanUpBucketByName(_sourceBucketName);
        await _s3ActionsWrapper.CleanUpBucketByName(_destinationBucketName);
    }
    else
    {
        Console.WriteLine(
            "Ok, we'll leave the resources intact.\n" +
            "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
        );
    }

    Console.WriteLine(new string('-', 80));
    return true;
}

```

```

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question)
    {
        Console.WriteLine(question);
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Helper method to get a choice response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="choices">The choices to print on the console.</param>
    /// <returns>The index of the selected choice</returns>
    private static int GetChoiceResponse(string? question, string[] choices, int
defaultChoice)
    {
        if (question != null)
        {
            Console.WriteLine(question);

            for (int i = 0; i < choices.Length; i++)
            {
                Console.WriteLine($"{i + 1}. {choices[i]}");
            }
        }

        if (!_interactive)
            return defaultChoice;

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.Length)
        {
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }
    }

```

```
    }

    return choiceNumber - 1;
}

/// <summary>
/// Get a string response from the user.
/// </summary>
/// <param name="question">The question to print.</param>
/// <param name="defaultAnswer">A default answer to use when not
interactive.</param>
/// <returns>The string response.</returns>
public static string GetStringResponse(string? question, string
defaultAnswer)
{
    string? answer = "";
    if (_interactive)
    {
        do
        {
            Console.WriteLine(question);
            answer = Console.ReadLine();
        } while (string.IsNullOrEmpty(answer));
    }
    else
    {
        answer = defaultAnswer;
    }

    return answer;
}
}
```

Una classe wrapper per le funzioni S3.

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Logging;

namespace S3ConditionalRequestsScenario;
```

```
/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;
    private readonly ILogger<S3ActionsWrapper> _logger;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    /// <param name="logger">The class logger.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, ILogger<S3ActionsWrapper> logger)
    {
        _amazonS3 = amazonS3;
        _logger = logger;
    }

    /// <summary>
    /// Retrieves an object from Amazon S3 with a conditional request.
    /// </summary>
    /// <param name="objectKey">The key of the object to retrieve.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="conditionType">The type of condition: 'IfMatch',
    'IfNoneMatch', 'IfModifiedSince', 'IfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
    dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
    etags.</param>
    /// <returns>True if the conditional read is successful, False otherwise.</
returns>
    public async Task<bool> GetObjectConditional(string objectKey, string
sourceBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
    {
        try
        {
            var getObjectRequest = new GetObjectRequest
            {
                BucketName = sourceBucket,
                Key = objectKey
            }
        }
    }
}
```

```
};

switch (conditionType)
{
    case S3ConditionType.IfMatch:
        getObjectRequest.EtagToMatch = etagConditionalValue;
        break;
    case S3ConditionType.IfNoneMatch:
        getObjectRequest.EtagToNotMatch = etagConditionalValue;
        break;
    case S3ConditionType.IfModifiedSince:
        getObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
        break;
    case S3ConditionType.IfUnmodifiedSince:
        getObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
        break;
    default:
        throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
}

var response = await _amazonS3.GetObjectAsync(getObjectRequest);
var sampleBytes = new byte[20];
await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
_logger.LogInformation($"Conditional read
successful. Here are the first 20 bytes of the object:
\n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
return true;
}
catch (AmazonS3Exception e)
{
    if (e.ErrorCode == "PreconditionFailed")
    {
        _logger.LogError("Conditional read failed: Precondition failed");
    }
    else if (e.ErrorCode == "NotModified")
    {
        _logger.LogError("Conditional read failed: Object not modified");
    }
    else
    {
        _logger.LogError($"Unexpected error: {e.ErrorCode}");
    }
}
```

```

        throw;
    }
    return false;
}

}

/// <summary>
/// Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite using an IfNoneMatch condition for the object key.
/// </summary>
/// <param name="objectKey">The key of the object to upload.</param>
/// <param name="bucket">The source bucket of the object.</param>
/// <param name="content">The content to upload as a string.</param>
/// <returns>The ETag if the conditional write is successful, empty
otherwise.</returns>
public async Task<string> PutObjectConditional(string objectKey, string
bucket, string content)
{
    try
    {
        var putObjectRequest = new PutObjectRequest
        {
            BucketName = bucket,
            Key = objectKey,
            ContentBody = content,
            IfNoneMatch = "*"
        };

        var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
        _logger.LogInformation($"Conditional write successful for key
{objectKey} in bucket {bucket}.");
        return putResult.ETag;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional write failed: Precondition
failed");
        }
        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
    }
}

```



```

    }
    return string.Empty;
}
}

/// <summary>
/// Copies an object from one Amazon S3 bucket to another with a conditional
request.
/// </summary>
/// <param name="sourceKey">The key of the source object to copy.</param>
/// <param name="destKey">The key of the destination object.</param>
/// <param name="sourceBucket">The source bucket of the object.</param>
/// <param name="destBucket">The destination bucket of the object.</param>
/// <param name="conditionType">The type of condition to apply, e.g.
'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
'CopySourceIfUnmodifiedSince'.</param>
/// <param name="conditionDateValue">The value to use for the condition for
dates.</param>
/// <param name="etagConditionalValue">The value to use for the condition for
etags.</param>
/// <returns>True if the conditional copy is successful, False otherwise.</
returns>
public async Task<bool> CopyObjectConditional(string sourceKey, string
destKey, string sourceBucket, string destBucket,
    S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
{
    try
    {
        var copyObjectRequest = new CopyObjectRequest
        {
            DestinationBucket = destBucket,
            DestinationKey = destKey,
            SourceBucket = sourceBucket,
            SourceKey = sourceKey
        };

        switch (conditionType)
        {
            case S3ConditionType.IfMatch:
                copyObjectRequest.ETagToMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfNoneMatch:
                copyObjectRequest.ETagToNotMatch = etagConditionalValue;

```

```

        break;
        case S3ConditionType.IfModifiedSince:
            copyObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
            break;
        case S3ConditionType.IfUnmodifiedSince:
            copyObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
            break;
        default:
            throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
    }

    await _amazonS3.CopyObjectAsync(copyObjectRequest);
    _logger.LogInformation($"Conditional copy successful for key
{destKey} in bucket {destBucket}.");
    return true;
}
catch (AmazonS3Exception e)
{
    if (e.ErrorCode == "PreconditionFailed")
    {
        _logger.LogError("Conditional copy failed: Precondition failed");
    }
    else if (e.ErrorCode == "304")
    {
        _logger.LogError("Conditional copy failed: Object not modified");
    }
    else
    {
        _logger.LogError($"Unexpected error: {e.ErrorCode}");
        throw;
    }
    return false;
}
}

/// <summary>
/// Create a new Amazon S3 bucket with a specified name and check that the
bucket is ready.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <returns>True if successful.</returns>

```

```

public async Task<bool> CreateBucketWithName(string bucketName)
{
    Console.WriteLine($"\\tCreating bucket {bucketName}.");
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true
        };

        await _amazonS3.PutBucketAsync(request);
        var bucketReady = false;
        var retries = 5;
        while (!bucketReady && retries > 0)
        {
            Thread.Sleep(5000);
            bucketReady = await
Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_amazonS3, bucketName);
            retries--;
        }

        return bucketReady;
    }
    catch (BucketAlreadyExistsException ex)
    {
        Console.WriteLine($"Bucket already exists: '{ex.Message}'");
        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
        return false;
    }
}

/// <summary>
/// Cleans up objects and deletes the bucket by name.
/// </summary>
/// <param name="bucketName">The name of the bucket.</param>
/// <returns>Async task.</returns>
public async Task CleanupBucketByName(string bucketName)
{
    try

```

```

        {
            var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
            foreach (var obj in listObjectsResponse.S3Objects)
            {
                await _amazonS3.DeleteObjectAsync(new DeleteObjectRequest
{ BucketName = bucketName, Key = obj.Key });
            }
            await _amazonS3.DeleteBucketAsync(new DeleteBucketRequest
{ BucketName = bucketName });
            Console.WriteLine($"Cleaned up bucket: {bucketName}.");
        }
        catch (AmazonS3Exception e)
        {
            if (e.ErrorCode == "NoSuchBucket")
            {
                Console.WriteLine($"Bucket {bucketName} does not exist, skipping
cleanup.");
            }
            else
            {
                Console.WriteLine($"Error deleting bucket: {e.ErrorCode}");
                throw;
            }
        }
    }
}

/// <summary>
/// List the contents of the bucket with their ETag.
/// </summary>
/// <param name="bucketName">The name of the bucket.</param>
/// <returns>Async task.</returns>
public async Task<List<S3Object>> ListBucketContentsByName(string bucketName)
{
    var results = new List<S3Object>();
    try
    {
        Console.WriteLine($"\\t Items in bucket {bucketName}");
        var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
        if (listObjectsResponse.S3Objects.Count == 0)
        {
            Console.WriteLine("\\t\\tNo objects found.");
        }
    }
}

```

```

        else
        {
            foreach (var obj in listObjectsResponse.S3Objects)
            {
                Console.WriteLine($"{obj.Key} ETag {obj.ETag}");
            }
        }
        results = listObjectsResponse.S3Objects;

    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "NoSuchBucket")
        {
            _logger.LogError($"Bucket {bucketName} does not exist.");
        }
        else
        {
            _logger.LogError($"Error listing bucket and objects:
{e.ErrorCode}");
            throw;
        }
    }

    return results;
}

/// <summary>
/// Delete an object from a specific bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <param name="objectKey">The key of the object to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteObjectFromBucket(string bucketName, string
objectKey)
{
    try
    {
        var request = new DeleteObjectRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };
        await _amazonS3.DeleteObjectAsync(request);
    }
}

```

```
        Console.WriteLine($"Deleted {objectKey} in {bucketName}.");
        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
        return false;
    }
}

/// <summary>
/// Delete a specific bucket by deleting the objects and then the bucket
itself.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <param name="objectKey">The key of the object to delete.</param>
/// <param name="versionId">Optional versionId.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CleanUpBucketByName(string bucketName)
{
    try
    {
        var allFiles = await ListBucketContentsByName(bucketName);

        foreach (var fileInfo in allFiles)
        {
            await DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key);
        }

        var request = new DeleteBucketRequest() { BucketName = bucketName, };
        var response = await _amazonS3.DeleteBucketAsync(request);
        Console.WriteLine($"\\tDelete for {bucketName} complete.");
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to delete bucket {bucketName}: " +
ex.Message);
        return false;
    }
}
```

```
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per .NET .
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Punto di ingresso per il flusso di lavoro (index.js). In questo modo vengono orchestrate tutte le fasi. Visita GitHub per vedere i dettagli di implementazione di Scenario ScenarioInput, ScenarioOutput, e ScenarioAction.

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "../welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
} from "../setup.steps.js";
```

```
/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new scenarios.Scenario(
      "S3 Conditional Requests - Deploy",
      [
        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        getBucketPrefix(scenarios),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        saveState,
      ],
      initialState,
    ),
    demo: new scenarios.Scenario(
      "S3 Conditional Requests - Demo",
      [loadState, welcome(scenarios), replAction(scenarios, client)],
      initialState,
    ),
    clean: new scenarios.Scenario(
      "S3 Conditional Requests - Destroy",
      [
        loadState,
        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
      ],
      initialState,
    ),
  };
};
```



```

};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const objectLockingScenarios = getWorkflowStages(Scenarios);
  Scenarios.parseScenarioArgs(objectLockingScenarios, {
    name: "Amazon S3 object locking workflow",
    description:
      "Work with Amazon Simple Storage Service (Amazon S3) object locking features.",
    synopsis:
      "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-v|--verbose]",
  });
}

```

Restituisce i messaggi di benvenuto alla console (welcome.steps.js).

```

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    "This example demonstrates the use of conditional requests for S3 operations." +
    " You can use conditional requests to add preconditions to S3 read requests to return " +
    "or copy an object based on its Entity tag (ETag), or last modified date.You can use " +
    "a conditional write requests to prevent overwrites by ensuring there is no existing " +
    "object with the same key.\n" +

```

```

    "This example will enable you to perform conditional reads and writes that
    will succeed " +
    "or fail based on your selected options.\n" +
    "Sample buckets and a sample object will be created as part of the example.
\n" +
    "Some steps require a key name prefix to be defined by the user. Before you
    begin, you can " +
    "optionally edit this prefix in ./object_name.json. If you do so, please
    reload the scenario before you begin.",
    { header: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };

```

Distribuisce bucket e oggetti (setup.steps.js).

```

import {
  ChecksumAlgorithm,
  CreateBucketCommand,
  PutObjectCommand,
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  S3ServiceException,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client

```

```

    */

    /**
     * @param {Scenarios} scenarios
     */
    const getBucketPrefix = (scenarios) =>
      new scenarios.ScenarioInput(
        "bucketPrefix",
        "Provide a prefix that will be used for bucket creation.",
        { type: "input", default: "amzn-s3-demo-bucket" },
      );

    /**
     * @param {Scenarios} scenarios
     */
    const createBuckets = (scenarios) =>
      new scenarios.ScenarioOutput(
        "createBuckets",
        (state) => `The following buckets will be created:
          ${state.bucketPrefix}-source-bucket.
          ${state.bucketPrefix}-destination-bucket.` ,
        { preformatted: true },
      );

    /**
     * @param {Scenarios} scenarios
     */
    const confirmCreateBuckets = (scenarios) =>
      new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
        type: "confirm",
      });

    /**
     * @param {Scenarios} scenarios
     * @param {S3Client} client
     */
    const createBucketsAction = (scenarios, client) =>
      new scenarios.ScenarioAction("createBucketsAction", async (state) => {
        const sourceBucketName = `${state.bucketPrefix}-source-bucket`;
        const destinationBucketName = `${state.bucketPrefix}-destination-bucket`;

        try {
          await client.send(
            new CreateBucketCommand({
              Bucket: sourceBucketName,

```

```

    }},
  );
  await waitUntilBucketExists({ client }, { Bucket: sourceBucketName });
  await client.send(
    new CreateBucketCommand({
      Bucket: destinationBucketName,
    }),
  );
  await waitUntilBucketExists(
    { client },
    { Bucket: destinationBucketName },
  );

  state.sourceBucketName = sourceBucketName;
  state.destinationBucketName = destinationBucketName;
} catch (caught) {
  if (
    caught instanceof BucketAlreadyExists ||
    caught instanceof BucketAlreadyOwnedByYou
  ) {
    console.error(`${caught.name}: ${caught.message}`);
    state.earlyExit = true;
  } else {
    throw caught;
  }
}
});

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",
    (state) => `The following test files will be created:
      file01.txt in ${state.bucketPrefix}-source-bucket.` ,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(

```

```
        "confirmPopulateBuckets",
        "Populate the buckets?",
        { type: "confirm" },
    );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
    new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
        try {
            await client.send(
                new PutObjectCommand({
                    Bucket: state.sourceBucketName,
                    Key: "file01.txt",
                    Body: "Content",
                    ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
                }),
            );
        } catch (caught) {
            if (caught instanceof S3ServiceException) {
                console.error(
                    `Error from S3 while uploading object.  ${caught.name}:
${caught.message}`,
                );
            } else {
                throw caught;
            }
        }
    });

export {
    confirmCreateBuckets,
    confirmPopulateBuckets,
    createBuckets,
    createBucketsAction,
    getBucketPrefix,
    populateBuckets,
    populateBucketsAction,
};
```

Ottiene, copia e inserisce oggetti utilizzando le richieste condizionali S3 (repl.steps.js).

```
import path from "node:path";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";

import {
  ListObjectVersionsCommand,
  GetObjectCommand,
  CopyObjectCommand,
  PutObjectCommand,
} from "@aws-sdk/client-s3";
import data from "../object_name.json" assert { type: "json" };
import { readFile } from "node:fs/promises";
import {
  ScenarioInput,
  Scenario,
  ScenarioAction,
  ScenarioOutput,
} from "../../libs/scenario/index.js";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  CONDITIONAL_READ: 2,
  CONDITIONAL_COPY: 3,
  CONDITIONAL_WRITE: 4,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new ScenarioInput(
    "replChoice",
```

```

    "Explore the S3 conditional request features by selecting one of the
    following choices",
    {
      type: "select",
      choices: [
        { name: "Print list of bucket items.", value: choices.LIST_ALL_FILES },
        {
          name: "Perform a conditional read.",
          value: choices.CONDITIONAL_READ,
        },
        {
          name: "Perform a conditional copy. These examples use the key name
          prefix defined in ./object_name.json.",
          value: choices.CONDITIONAL_COPY,
        },
        {
          name: "Perform a conditional write. This example use the sample file ./
          text02.txt.",
          value: choices.CONDITIONAL_WRITE,
        },
        { name: "Finish the workflow.", value: choices.EXIT },
      ],
    },
  );

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
  /** @type {{bucket: string, key: string, version: string}[]} */
  const files = [];
  for (const bucket of buckets) {
    const objectsResponse = await client.send(
      new ListObjectVersionsCommand({ Bucket: bucket }),
    );
    for (const version of objectsResponse.Versions || []) {
      const { Key } = version;
      files.push({ bucket, key: Key });
    }
  }
  return files;
};

```

```
/**
 * @param {S3Client} client
 * @param {string[]} buckets
 * @param {string} key
 */
const getEtag = async (client, bucket, key) => {
  const objectsResponse = await client.send(
    new GetObjectCommand({
      Bucket: bucket,
      Key: key,
    }),
  );
  return objectsResponse.ETag;
};

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
export const replAction = (scenarios, client) =>
  new ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
        state.sourceBucketName,
        state.destinationBucketName,
      ]);

      const fileInput = new scenarios.ScenarioInput(
        "selectedFile",
        "Select a file to use:",
        {
          type: "select",
          choices: files.map((file, index) => ({
            name: `${index + 1}: ${file.bucket}: ${file.key} (Etag: ${
              file.version
            })`,
            value: index,
          })),
        },
      );
    },
  );
```



```
    },
  );
  const condReadOptions = new scenarios.ScenarioInput(
    "selectOption",
    "Which conditional read action would you like to take?",
    {
      type: "select",
      choices: [
        "If-Match: using the object's ETag. This condition should succeed.",
        "If-None-Match: using the object's ETag. This condition should
fail.",
        "If-Modified-Since: using yesterday's date. This condition should
succeed.",
        "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
      ],
    },
  );
  const condCopyOptions = new scenarios.ScenarioInput(
    "selectOption",
    "Which conditional copy action would you like to take?",
    {
      type: "select",
      choices: [
        "If-Match: using the object's ETag. This condition should succeed.",
        "If-None-Match: using the object's ETag. This condition should
fail.",
        "If-Modified-Since: using yesterday's date. This condition should
succeed.",
        "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
      ],
    },
  );
  const condWriteOptions = new scenarios.ScenarioInput(
    "selectOption",
    "Which conditional write action would you like to take?",
    {
      type: "select",
      choices: [
        "IfNoneMatch condition on the object key: If the key is a duplicate,
the write will fail.",
      ],
    },
  ),
```

```

    );

    const { replChoice } = state;

    switch (replChoice) {
      case choices.LIST_ALL_FILES: {
        const files = await getAllFiles(client, [
          state.sourceBucketName,
          state.destinationBucketName,
        ]);
        state.replOutput = files
          .map(
            (file) => `Items in bucket ${file.bucket}: object: ${file.key} `,
          )
          .join("\n");
        break;
      }
      case choices.CONDITIONAL_READ:
        {
          const selectedCondRead = await condReadOptions.handle(state);
          if (
            selectedCondRead ===
            "If-Match: using the object's ETag. This condition should succeed."
          ) {
            const bucket = state.sourceBucketName;
            const key = "file01.txt";
            const ETag = await getEtag(client, bucket, key);

            try {
              await client.send(
                new GetObjectCommand({
                  Bucket: bucket,
                  Key: key,
                  IfMatch: ETag,
                }),
              );
              state.replOutput = `${key} in bucket ${state.sourceBucketName}
read because ETag provided matches the object's ETag.`;
            } catch (err) {
              state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
            }
            break;
          }
        }
    }
  }

```

```
        if (
            selectedCondRead ===
            "If-None-Match: using the object's ETag. This condition should
fail."
        ) {
            const bucket = state.sourceBucketName;
            const key = "file01.txt";
            const ETag = await getEtag(client, bucket, key);

            try {
                await client.send(
                    new GetObjectCommand({
                        Bucket: bucket,
                        Key: key,
                        IfNoneMatch: ETag,
                    }),
                );
                state.replOutput = `${key} in ${state.sourceBucketName} was
returned.`;
            } catch (err) {
                state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
            }
            break;
        }
        if (
            selectedCondRead ===
            "If-Modified-Since: using yesterday's date. This condition should
succeed."
        ) {
            const date = new Date();
            date.setDate(date.getDate() - 1);

            const bucket = state.sourceBucketName;
            const key = "file01.txt";
            try {
                await client.send(
                    new GetObjectCommand({
                        Bucket: bucket,
                        Key: key,
                        IfModifiedSince: date,
                    }),
                );
            }
```

```

        state.replOutput = `${key} in bucket ${state.sourceBucketName}
read because it has been created or modified in the last 24 hours.`;
    } catch (err) {
        state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
    }
    break;
}
if (
    selectedCondRead ===
    "If-Unmodified-Since: using yesterday's date. This condition should
fail."
) {
    const bucket = state.sourceBucketName;
    const key = "file01.txt";

    const date = new Date();
    date.setDate(date.getDate() - 1);
    try {
        await client.send(
            new GetObjectCommand({
                Bucket: bucket,
                Key: key,
                IfUnmodifiedSince: date,
            }),
        );
        state.replOutput = `${key} in ${state.sourceBucketName} was
read.`;
    } catch (err) {
        state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
    }
    break;
}
break;
case choices.CONDITIONAL_COPY: {
    const selectedCondCopy = await condCopyOptions.handle(state);
    if (
        selectedCondCopy ===
        "If-Match: using the object's ETag. This condition should succeed."
    ) {
        const bucket = state.sourceBucketName;
        const key = "file01.txt";

```

```

    const ETag = await getEtag(client, bucket, key);

    const copySource = `${bucket}/${key}`;
    // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
    const name = data.name;
    const copiedKey = `${name}${key}`;
    try {
      await client.send(
        new CopyObjectCommand({
          CopySource: copySource,
          Bucket: state.destinationBucketName,
          Key: copiedKey,
          CopySourceIfMatch: ETag,
        }),
      );
      state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because ETag provided matches the object's
ETag.`;
    } catch (err) {
      state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName}: ${err.message}`;
    }
    break;
  }
  if (
    selectedCondCopy ===
    "If-None-Match: using the object's ETag. This condition should fail."
  ) {
    const bucket = state.sourceBucketName;
    const key = "file01.txt";
    const ETag = await getEtag(client, bucket, key);
    const copySource = `${bucket}/${key}`;
    // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
    const name = data.name;
    const copiedKey = `${name}${key}`;

    try {
      await client.send(
        new CopyObjectCommand({
          CopySource: copySource,
          Bucket: state.destinationBucketName,
          Key: copiedKey,

```

```

        CopySourceIfNoneMatch: ETag,
    )),
    );
    state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName}`;
    } catch (err) {
        state.replOutput = `Unable to copy object as ${key} as as
${copiedKey} to bucket ${state.destinationBucketName}: ${err.message}`;
    }
    break;
}
if (
    selectedCondCopy ===
    "If-Modified-Since: using yesterday's date. This condition should
succeed."
) {
    const bucket = state.sourceBucketName;
    const key = "file01.txt";
    const copySource = `${bucket}/${key}`;
    // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
    const name = data.name;
    const copiedKey = `${name}${key}`;

    const date = new Date();
    date.setDate(date.getDate() - 1);

    try {
        await client.send(
            new CopyObjectCommand({
                CopySource: copySource,
                Bucket: state.destinationBucketName,
                Key: copiedKey,
                CopySourceIfModifiedSince: date,
            )),
        );
        state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because it has been created or modified in the
last 24 hours.`;
    } catch (err) {
        state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName} : ${err.message}`;
    }
    break;
}

```

```

    }
    if (
      selectedCondCopy ===
        "If-Unmodified-Since: using yesterday's date. This condition should
fail."
    ) {
      const bucket = state.sourceBucketName;
      const key = "file01.txt";
      const copySource = `${bucket}/${key}`;
      // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
      const name = data.name;
      const copiedKey = `${name}${key}`;

      const date = new Date();
      date.setDate(date.getDate() - 1);

      try {
        await client.send(
          new CopyObjectCommand({
            CopySource: copySource,
            Bucket: state.destinationBucketName,
            Key: copiedKey,
            CopySourceIfUnmodifiedSince: date,
          }),
        );
        state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName} because it has not been created or modified in
the last 24 hours.`;
      } catch (err) {
        state.replOutput = `Unable to copy object ${key} to bucket
${state.destinationBucketName}: ${err.message}`;
      }
    }
    break;
  }
  case choices.CONDITIONAL_WRITE:
    {
      const selectedCondWrite = await condWriteOptions.handle(state);
      if (
        selectedCondWrite ===
          "IfNoneMatch condition on the object key: If the key is a
duplicate, the write will fail."
      ) {

```

```

        // Optionally edit the default key name prefix of the copied object
        in ./object_name.json.
        const key = "text02.txt";
        const __filename = fileURLToPath(import.meta.url);
        const __dirname = dirname(__filename);
        const filePath = path.join(__dirname, "text02.txt");
        try {
            await client.send(
                new PutObjectCommand({
                    Bucket: `${state.destinationBucketName}`,
                    Key: `${key}`,
                    Body: await readFile(filePath),
                    IfNoneMatch: "*",
                }),
            );
            state.replOutput = `${key} uploaded to bucket
${state.destinationBucketName} because the key is not a duplicate.`;
        } catch (err) {
            state.replOutput = `Unable to upload object to bucket
${state.destinationBucketName}:${err.message}`;
        }
        break;
    }
}
break;

default:
    throw new Error(`Invalid replChoice: ${replChoice}`);
}
},
{
    whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new ScenarioOutput("REPL output", (state) => state.replOutput, {
            preformatted: true,
        }),
    },
},
);

export { replInput, choices };

```


Elimina tutte le risorse create (clean.steps.js).

```
import {
  DeleteObjectCommand,
  DeleteBucketCommand,
  ListObjectVersionsCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { sourceBucketName, destinationBucketName } = state;
    const buckets = [sourceBucketName, destinationBucketName].filter((b) => b);

    for (const bucket of buckets) {
      try {
        let objectsResponse;
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
        for (const version of objectsResponse.Versions || []) {
          const { Key, VersionId } = version;
          try {
```

```
        await client.send(
            new DeleteObjectCommand({
                Bucket: bucket,
                Key,
                VersionId,
            }),
        );
    } catch (err) {
        console.log(`An error occurred: ${err.message} `);
    }
}
} catch (e) {
    if (e instanceof Error && e.name === "NoSuchBucket") {
        console.log("Objects and buckets have already been deleted.");
        continue;
    }
    throw e;
}

await client.send(new DeleteBucketCommand({ Bucket: bucket }));
console.log(`Delete for ${bucket} complete.`);
}
});

export { confirmCleanup, cleanupAction };
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per JavaScript .
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

Python

SDK per Python (Boto3)

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegui uno scenario interattivo che dimostra le richieste condizionali di Amazon S3.

```
"""
Purpose

Shows how to use AWS SDK for Python (Boto3) to get started using conditional
requests for
Amazon Simple Storage Service (Amazon S3).

"""

import logging
import random
import sys
import datetime

import boto3
from botocore.exceptions import ClientError

from s3_conditional_requests import S3ConditionalRequests

# Add relative path to include demo_tools in this code example without need for
# setup.
sys.path.append("../..")
import demo_tools.question as q # noqa

# Constants
FILE_CONTENT = "This is a test file for S3 conditional requests."
RANDOM_SUFFIX = str(random.randint(100, 999))

logger = logging.getLogger(__name__)
```

```
class ConditionalRequestsScenario:
    """Runs a scenario that shows how to use S3 Conditional Requests."""

    def __init__(self, conditional_requests, s3_client):
        """
        :param conditional_requests: An object that wraps S3 conditional request
        actions.
        :param s3_client: A Boto3 S3 client for setup and cleanup operations.
        """
        self.conditional_requests = conditional_requests
        self.s3_client = s3_client

    def setup_scenario(self, source_bucket: str, dest_bucket: str, object_key:
    str):
        """
        Sets up the scenario by creating a source and destination bucket.
        Prompts the user to provide a bucket name prefix.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        :param object_key: The name of a test file to add to the source bucket.
        """

        # Create the buckets.
        try:
            self.s3_client.create_bucket(Bucket=source_bucket)
            self.s3_client.create_bucket(Bucket=dest_bucket)
            print(
                f"Created source bucket: {source_bucket} and destination bucket:
{dest_bucket}"
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            logger.error(f"Error creating buckets: {error_code}")
            raise

        # Upload test file into the source bucket.
        try:
            print(f"Uploading file {object_key} to bucket {source_bucket}")
            response = self.s3_client.put_object(
                Bucket=source_bucket, Key=object_key, Body=FILE_CONTENT
            )
            object_etag = response["ETag"]
            return object_etag
```

```

        except Exception as e:
            logger.error(
                f"Failed to upload file {object_key} to bucket {source_bucket}:"
                {e}"
            )

def cleanup_scenario(self, source_bucket: str, dest_bucket: str):
    """
    Cleans up the scenario by deleting the source and destination buckets.

    :param source_bucket: The name of the source bucket.
    :param dest_bucket: The name of the destination bucket.
    """
    self.cleanup_bucket(source_bucket)
    self.cleanup_bucket(dest_bucket)

def cleanup_bucket(self, bucket_name: str):
    """
    Cleans up the bucket by deleting all objects and then the bucket itself.

    :param bucket_name: The name of the bucket.
    """
    try:
        # Get list of all objects in the bucket.
        list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
        objs = list_response.get("Contents", [])
        for obj in objs:
            key = obj["Key"]
            self.s3_client.delete_object(Bucket=bucket_name, Key=key)
        self.s3_client.delete_bucket(Bucket=bucket_name)
        print(f"Cleaned up bucket: {bucket_name}.")
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "NoSuchBucket":
            logger.info(f"Bucket {bucket_name} does not exist, skipping
cleanup.")
        else:
            logger.error(f"Error deleting bucket: {error_code}")
            raise

def display_buckets(self, source_bucket: str, dest_bucket: str):

```

```

    """
    Display a list of the objects in the test buckets.

    :param source_bucket: The name of the source bucket.
    :param dest_bucket: The name of the destination bucket.
    """
    self.list_bucket_contents(source_bucket)
    self.list_bucket_contents(dest_bucket)

def list_bucket_contents(self, bucket_name):
    """
    Display a list of the objects in the bucket.

    :param bucket_name: The name of the bucket.
    """
    try:
        # Get list of all objects in the bucket.
        print(f"\t Items in bucket {bucket_name}")
        list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
        objs = list_response.get("Contents", [])
        if not objs:
            print("\t\tNo objects found.")
        for obj in objs:
            key = obj["Key"]
            print(f"\t\t object: {key} ETag {obj['ETag']}")
        return objs
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "NoSuchBucket":
            logger.info(f"Bucket {bucket_name} does not exist.")
        else:
            logger.error(f"Error listing bucket and objects: {error_code}")
            raise

def display_menu(
    self, source_bucket: str, dest_bucket: str, object_key: str, etag: str
):
    """
    Displays the menu of conditional request options for the user.

    :param source_bucket: The name of the source bucket.
    :param dest_bucket: The name of the destination bucket.
    :param object_key: The key of the test object in the source bucket.

```

```

:param etag: The etag of the test object in the source bucket.
"""

actions = [
    "Print list of bucket items.",
    "Perform a conditional read.",
    "Perform a conditional copy.",
    "Perform a conditional write.",
    "Clean up and exit.",
]

conditions = [
    "If-Match: using the object's ETag. This condition should succeed.",
    "If-None-Match: using the object's ETag. This condition should
fail.",
    "If-Modified-Since: using yesterday's date. This condition should
succeed.",
    "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
]

condition_types = [
    "IfMatch",
    "IfNoneMatch",
    "IfModifiedSince",
    "IfUnmodifiedSince",
]

copy_condition_types = [
    "CopySourceIfMatch",
    "CopySourceIfNoneMatch",
    "CopySourceIfModifiedSince",
    "CopySourceIfUnmodifiedSince",
]

yesterday_date = datetime.datetime.utcnow() - datetime.timedelta(days=1)

choice = 0
while choice != 4:
    print("-" * 88)
    print("Choose an action to explore some example conditional
requests.")
    choice = q.choose("Which action would you like to take? ", actions)
    if choice == 0:
        print("Listing the objects and buckets.")

```

```

        self.display_buckets(source_bucket, dest_bucket)
    elif choice == 1:
        print("Perform a conditional read.")
        condition_type = q.choose("Enter the condition type : ",
conditions)
        if condition_type == 0 or condition_type == 1:
            self.conditional_requests.get_object_conditional(
                object_key, source_bucket,
conditions_types[condition_type], etag
            )
        elif condition_type == 2 or condition_type == 3:
            self.conditional_requests.get_object_conditional(
                object_key,
                source_bucket,
                condition_types[condition_type],
                yesterday_date,
            )
    elif choice == 2:
        print("Perform a conditional copy.")
        condition_type = q.choose("Enter the condition type : ",
conditions)
        dest_key = q.ask("Enter an object key: ", q.non_empty)
        if condition_type == 0 or condition_type == 1:
            self.conditional_requests.copy_object_conditional(
                object_key,
                dest_key,
                source_bucket,
                dest_bucket,
                copy_condition_types[condition_type],
                etag,
            )
        elif condition_type == 2 or condition_type == 3:
            self.conditional_requests.copy_object_conditional(
                object_key,
                dest_key,
                copy_condition_types[condition_type],
                yesterday_date,
            )
    elif choice == 3:
        print(
            "Perform a conditional write using IfNoneMatch condition on
the object key."
        )
        print("If the key is a duplicate, the write will fail.")

```



```

        object_key = q.ask("Enter an object key: ", q.non_empty)
        self.conditional_requests.put_object_conditional(
            object_key, source_bucket, b"Conditional write example data."
        )
    elif choice == 4:
        print("Proceeding to cleanup.")

def run_scenario(self):
    """
    Runs the interactive scenario.
    """
    print("-" * 88)
    print("Welcome to the Amazon S3 conditional requests example.")
    print("-" * 88)

    print(
        f"""\
This example demonstrates the use of conditional requests for S3
operations.
You can use conditional requests to add preconditions to S3 read requests
to return or copy
an object based on its Entity tag (ETag), or last modified date.
You can use a conditional write requests to prevent overwrites by
ensuring
there is no existing object with the same key.

This example will allow you to perform conditional reads
and writes that will succeed or fail based on your selected options.

Sample buckets and a sample object will be created as part of the
example.
"""
    )

    bucket_prefix = q.ask("Enter a bucket name prefix: ", q.non_empty)
    source_bucket_name = f"{bucket_prefix}-source-{RANDOM_SUFFIX}"
    dest_bucket_name = f"{bucket_prefix}-dest-{RANDOM_SUFFIX}"
    object_key = "test-upload-file.txt"

    try:
        etag = self.setup_scenario(source_bucket_name, dest_bucket_name,
            object_key)

```

```
        self.display_menu(source_bucket_name, dest_bucket_name, object_key,
                           etag)
        finally:
            self.cleanup_scenario(source_bucket_name, dest_bucket_name)

        print("-" * 88)
        print("Thanks for watching.")
        print("-" * 88)

if __name__ == "__main__":
    scenario = ConditionalRequestsScenario(
        S3ConditionalRequests.from_client(), boto3.client("s3")
    )
    scenario.run_scenario()
```

Una classe wrapper che definisce le operazioni delle richieste condizionali.

```
import boto3
import logging

from botocore.exceptions import ClientError

# Configure logging
logger = logging.getLogger(__name__)

class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)
```

```

def get_object_conditional(
    self,
    object_key: str,
    source_bucket: str,
    condition_type: str,
    condition_value: str,
):
    """
    Retrieves an object from Amazon S3 with a conditional request.

    :param object_key: The key of the object to retrieve.
    :param source_bucket: The source bucket of the object.
    :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
    'IfModifiedSince', 'IfUnmodifiedSince'.
    :param condition_value: The value to use for the condition.
    """
    try:
        response = self.s3.get_object(
            Bucket=source_bucket,
            Key=object_key,
            **{condition_type: condition_value},
        )
        sample_bytes = response["Body"].read(20)
        print(
            f"\tConditional read successful. Here are the first 20 bytes of
the object:\n"
        )
        print(f"\t{sample_bytes}")
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "PreconditionFailed":
            print("\tConditional read failed: Precondition failed")
        elif error_code == "304": # Not modified error code.
            print("\tConditional read failed: Object not modified")
        else:
            logger.error(f"Unexpected error: {error_code}")
            raise

def put_object_conditional(self, object_key: str, source_bucket: str, data:
bytes):

```

```

    """
    Uploads an object to Amazon S3 with a conditional request. Prevents
    overwrite
    using an IfNoneMatch condition for the object key.

    :param object_key: The key of the object to upload.
    :param source_bucket: The source bucket of the object.
    :param data: The data to upload.
    """
    try:
        self.s3.put_object(
            Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch="*"
        )
        print(
            f"\tConditional write successful for key {object_key} in bucket
{source_bucket}."
        )
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "PreconditionFailed":
            print("\tConditional write failed: Precondition failed")
        else:
            logger.error(f"Unexpected error: {error_code}")
            raise

def copy_object_conditional(
    self,
    source_key: str,
    dest_key: str,
    source_bucket: str,
    dest_bucket: str,
    condition_type: str,
    condition_value: str,
):
    """
    Copies an object from one Amazon S3 bucket to another with a conditional
    request.

    :param source_key: The key of the source object to copy.
    :param dest_key: The key of the destination object.
    :param source_bucket: The source bucket of the object.
    :param dest_bucket: The destination bucket of the object.
    :param condition_type: The type of condition to apply, e.g.

```

```

    'CopySourceIfMatch', 'CopySourceIfNoneMatch',
    'CopySourceIfModifiedSince', 'CopySourceIfUnmodifiedSince'.
    :param condition_value: The value to use for the condition.
    """
    try:
        self.s3.copy_object(
            Bucket=dest_bucket,
            Key=dest_key,
            CopySource={"Bucket": source_bucket, "Key": source_key},
            **{condition_type: condition_value},
        )
        print(
            f"\tConditional copy successful for key {dest_key} in bucket {dest_bucket}."
        )
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "PreconditionFailed":
            print("\tConditional copy failed: Precondition failed")
        elif error_code == "304": # Not modified error code.
            print("\tConditional copy failed: Object not modified")
        else:
            logger.error(f"Unexpected error: {error_code}")
            raise

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Python (Boto3).
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Gestisci gli elenchi di controllo degli accessi (ACL) per i bucket Amazon S3 utilizzando un AWS SDK

Gli esempi di codice seguenti mostrano come gestire le liste di controllo degli accessi (ACL) per i bucket Amazon S3.

.NET

SDK per .NET

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to manage Amazon Simple Storage Service
/// (Amazon S3) access control lists (ACLs) to control Amazon S3 bucket
/// access.
/// </summary>
public class ManageACLs
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket1";
        string newBucketName = "amzn-s3-demo-bucket2";
        string keyName = "sample-object.txt";
        string emailAddress = "someone@example.com";

        // If the AWS Region where your bucket is located is different from
        // the Region defined for the default user, pass the Amazon S3
        bucket's
        // name to the client constructor. It should look like this:
        // RegionEndpoint bucketRegion = RegionEndpoint.USEast1;
        IAmazonS3 client = new AmazonS3Client();
```

```

        await TestBucketObjectACLsAsync(client, bucketName, newBucketName,
keyName, emailAddress);
    }

    /// <summary>
    /// Creates a new Amazon S3 bucket with a canned ACL, then retrieves the
ACL
    /// information and then adds a new ACL to one of the objects in the
    /// Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// methods to create a bucket, get an ACL, and add a different ACL to
    /// one of the objects.</param>
    /// <param name="bucketName">A string representing the original Amazon S3
    /// bucket name.</param>
    /// <param name="newBucketName">A string representing the name of the
    /// new bucket that will be created.</param>
    /// <param name="keyName">A string representing the key name of an Amazon
S3
    /// object for which we will change the ACL.</param>
    /// <param name="emailAddress">A string representing the email address
    /// belonging to the person to whom access to the Amazon S3 bucket will
be
    /// granted.</param>
    public static async Task TestBucketObjectACLsAsync(
        IAmazonS3 client,
        string bucketName,
        string newBucketName,
        string keyName,
        string emailAddress)
    {
        try
        {
            // Create a new Amazon S3 bucket and specify canned ACL.
            var success = await CreateBucketWithCannedACLAsync(client,
newBucketName);

            // Get the ACL on a bucket.
            await GetBucketACLAsync(client, bucketName);

            // Add (replace) the ACL on an object in a bucket.

```

```

        await AddACLToExistingObjectAsync(client, bucketName, keyName,
emailAddress);
    }
    catch (AmazonS3Exception amazonS3Exception)
    {
        Console.WriteLine($"Exception: {amazonS3Exception.Message}");
    }
}

/// <summary>
/// Creates a new Amazon S3 bucket with a canned ACL attached.
/// </summary>
/// <param name="client">The initialized client object used to call
/// PutBucketAsync.</param>
/// <param name="newBucketName">A string representing the name of the
/// new Amazon S3 bucket.</param>
/// <returns>Returns a boolean value indicating success or failure.</
returns>
public static async Task<bool> CreateBucketWithCannedACLAsync(IAmazonS3
client, string newBucketName)
{
    var request = new PutBucketRequest()
    {
        BucketName = newBucketName,
        BucketRegion = S3Region.EUWest1,

        // Add a canned ACL.
        CannedACL = S3CannedACL.LogDeliveryWrite,
    };

    var response = await client.PutBucketAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Retrieves the ACL associated with the Amazon S3 bucket name in the
/// bucketName parameter.
/// </summary>
/// <param name="client">The initialized client object used to call
/// PutBucketAsync.</param>
/// <param name="bucketName">The Amazon S3 bucket for which we want to
get the
/// ACL list.</param>

```



```

    /// <returns>Returns an S3AccessControlList returned from the call to
    /// GetACLAsync.</returns>
    public static async Task<S3AccessControlList> GetBucketACLAsync(IAmazonS3
client, string bucketName)
    {
        GetACLResponse response = await client.GetACLAsync(new GetACLRequest
        {
            BucketName = bucketName,
        });

        return response.AccessControlList;
    }

    /// <summary>
    /// Adds a new ACL to an existing object in the Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized client object used to call
    /// PutBucketAsync.</param>
    /// <param name="bucketName">A string representing the name of the Amazon
S3
param>
    /// bucket containing the object to which we want to apply a new ACL.</
param>
    /// <param name="keyName">A string representing the name of the object
    /// to which we want to apply the new ACL.</param>
    /// <param name="emailAddress">The email address of the person to whom
    /// we will be applying to whom access will be granted.</param>
    public static async Task AddACLToExistingObjectAsync(IAmazonS3 client,
string bucketName, string keyName, string emailAddress)
    {
        // Retrieve the ACL for an object.
        GetACLResponse aclResponse = await client.GetACLAsync(new
GetACLRequest
        {
            BucketName = bucketName,
            Key = keyName,
        });

        S3AccessControlList acl = aclResponse.AccessControlList;

        // Retrieve the owner.
        Owner owner = acl.Owner;

```

```
// Clear existing grants.
acl.Grants.Clear();

// Add a grant to reset the owner's full permission
// (the previous clear statement removed all permissions).
var fullControlGrant = new S3Grant
{
    Grantee = new S3Grantee { CanonicalUser = acl.Owner.Id },
};
acl.AddGrant(fullControlGrant.Grantee, S3Permission.FULL_CONTROL);

// Specify email to identify grantee for granting permissions.
var grantUsingEmail = new S3Grant
{
    Grantee = new S3Grantee { EmailAddress = emailAddress },
    Permission = S3Permission.WRITE_ACP,
};

// Specify log delivery group as grantee.
var grantLogDeliveryGroup = new S3Grant
{
    Grantee = new S3Grantee { URI = "http://acs.amazonaws.com/groups/
s3/LogDelivery" },
    Permission = S3Permission.WRITE,
};

// Create a new ACL.
var newAcl = new S3AccessControlList
{
    Grants = new List<S3Grant> { grantUsingEmail,
grantLogDeliveryGroup },
    Owner = owner,
};

// Set the new ACL. We're throwing away the response here.
_ = await client.PutACLAsync(new PutACLRequest
{
    BucketName = bucketName,
    Key = keyName,
    AccessControlList = newAcl,
});
}

}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per .NET .
 - [GetBucketAcl](#)
 - [GetObjectAcl](#)
 - [PutBucketAcl](#)
 - [PutObjectAcl](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Gestisci messaggi Amazon SQS di grandi dimensioni utilizzando Amazon S3 con un AWS SDK

L'esempio di codice seguente mostra come utilizzare la libreria client ampia di Amazon SQS utilizzare messaggi Amazon SQS di grandi dimensioni.

Java

SDK per Java 2.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
```

```
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
import java.util.List;
import java.util.UUID;

/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
        LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
```

```
        this(S3Client.create(), 300000);
    }

    /**
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
            DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());

        // Generate a unique queue name.
        this.queueName = "MyQueue-" + UUID.randomUUID();

        // Configure the SQS extended client.
        final ExtendedClientConfiguration extendedClientConfig = new
        ExtendedClientConfiguration()
            .withPayloadSupportEnabled(s3Client, s3BucketName);

        this.sqsExtendedClient = new
        AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
    }

    public static void main(String[] args) {
        SqsExtendedClientExample example = new SqsExtendedClientExample();
        try {
            example.setup();
            example.sendAndReceiveMessage();
        } finally {
            example.cleanup();
        }
    }

    /**
     * Send a large message and receive it back.
     *
     * @return The received message
     */
```

```
public Message sendAndReceiveMessage() {
    try {
        // Create a large message.
        char[] chars = new char[messageSize];
        Arrays.fill(chars, 'x');
        String largeMessage = new String(chars);

        // Send the message.
        final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
            .queueUrl(queueUrl)
            .messageBody(largeMessage)
            .build();

        sqsExtendedClient.sendMessage(sendMessageRequest);
        logger.info("Sent message of size: {}", largeMessage.length());

        // Receive and return the message.
        final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
            ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

        List<Message> messages = receiveMessageResponse.messages();
        if (messages.isEmpty()) {
            throw new RuntimeException("No messages received");
        }

        Message message = messages.getFirst();
        logger.info("\nMessage received.");
        logger.info("  ID: {}", message.messageId());
        logger.info("  Receipt handle: {}", message.receiptHandle());
        logger.info("  Message body size: {}", message.body().length());
        logger.info("  Message body (first 5 characters): {}",
message.body().substring(0, 5));

        return message;
    } catch (RuntimeException e) {
        logger.error("Error during message processing: {}", e.getMessage(),
e);
        throw e;
    }
}
```

- Per ulteriori informazioni, consulta la [Guida per sviluppatori di AWS SDK for Java 2.x](#).
- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK for Java 2.x .
 - [CreateBucket](#)
 - [PutBucketLifecycleConfiguration](#)
 - [ReceiveMessage](#)
 - [SendMessage](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Gestisci oggetti Amazon S3 con versioni in batch con una funzione Lambda utilizzando un AWS SDK

L'esempio di codice seguente mostra come gestire gli oggetti con versione S3 in batch con una funzione Lambda.

Python

SDK per Python (Boto3)

Mostra come manipolare oggetti con versione di Amazon Simple Storage Service (Amazon S3) in batch creando processi che richiamano funzioni per eseguire l'elaborazione. AWS Lambda Questo esempio mostra come creare un bucket abilitato per le versioni, caricare le strofe dalla poesia You Are Old, Father William di Lewis Carroll e utilizzare i processi batch Amazon S3 per eseguire varie operazioni sulla poesia.

Scopri come:

- Creare funzioni Lambda che operano su oggetti con versione.
- Creare un manifesto di oggetti da aggiornare.
- Creare processi batch che invocano le funzioni Lambda per aggiornare gli oggetti.
- Eliminare funzioni Lambda.
- Svuotare ed eliminare un bucket con versione.

Questo esempio è visualizzato al meglio su. GitHub Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, vedi l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- Simple Storage Service (Amazon S3)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Analizza gli URI di Amazon S3 utilizzando un AWS SDK

L'esempio di codice seguente mostra come analizzare gli URI di Amazon S3 per estrarre componenti importanti, come il nome del bucket e la chiave dell'oggetto.

Java

SDK per Java 2.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Analizza un URI Amazon S3 utilizzando la classe [S3Uri](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
import java.util.Map;

/**
 *
 * @param s3Client - An S3Client through which you acquire an S3Uri
 instance.
 * @param s3ObjectUrl - A complex URL (String) that is used to demonstrate
 S3Uri
```



```
*                capabilities.
*/
public static void parseS3UriExample(S3Client s3Client, String s3objectUrl) {
    logger.info(s3objectUrl);
    // Console output:
    // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
versionId=abc123&partNumber=77&partNumber=88'.

    // Create an S3Utilities object using the configuration of the s3Client.
    S3Utilities s3Utilities = s3Client.utilities();

    // From a String URL create a URI object to pass to the parseUri()
method.
    URI uri = URI.create(s3objectUrl);
    S3Uri s3Uri = s3Utilities.parseUri(uri);

    // If the URI contains no value for the Region, bucket or key, the SDK
returns
    // an empty Optional.
    // The SDK returns decoded URI values.

    Region region = s3Uri.region().orElse(null);
    log("region", region);
    // Console output: 'region: us-west-1'.

    String bucket = s3Uri.bucket().orElse(null);
    log("bucket", bucket);
    // Console output: 'bucket: myBucket'.

    String key = s3Uri.key().orElse(null);
    log("key", key);
    // Console output: 'key: resources/doc.txt'.

    Boolean isPathStyle = s3Uri.isPathStyle();
    log("isPathStyle", isPathStyle);
    // Console output: 'isPathStyle: true'.

    // If the URI contains no query parameters, the SDK returns an empty map.
    Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
    log("rawQueryParameters", queryParams);
    // Console output: 'rawQueryParameters: {versionId=[abc123],
partNumber=[77,
    // 88]}'.
}
```

```

        // Retrieve the first or all values for a query parameter as shown in the
        // following code.
        String versionId =
s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
        log("firstMatchingRawQueryParameter-versionId", versionId);
        // Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.

        String partNumber =
s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
        log("firstMatchingRawQueryParameter-partNumber", partNumber);
        // Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

        List<String> partNumbers =
s3Uri.firstMatchingRawQueryParameters("partNumber");
        log("firstMatchingRawQueryParameter", partNumbers);
        // Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

        /*
         * Object keys and query parameters with reserved or unsafe characters,
must be
         * URL-encoded.
         * For example replace whitespace " " with "%20".
         * Valid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
%5Bbrackets%5D"
         * Invalid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object key?
query=[brackets]"
         *
         * Virtual-hosted-style URIs with bucket names that contain a dot, ".",
the dot
         * must not be URL-encoded.
         * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
         * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
         */
    }

    private static void log(String s3UriElement, Object element) {
        if (element == null) {
            logger.info("{}: {}", s3UriElement, "null");
        } else {
            logger.info("{}: {}", s3UriElement, element);
        }
    }
}

```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Esegui una copia multiparte di un oggetto Amazon S3 utilizzando un AWS SDK

L'esempio di codice seguente mostra come eseguire una copia in più parti di un oggetto Amazon S3.

.NET

SDK per .NET

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to perform a multi-part copy from one Amazon
/// Simple Storage Service (Amazon S3) bucket to another.
/// </summary>
public class MPUapiCopyObj
{
    private const string SourceBucket = "amzn-s3-demo-bucket1";
    private const string TargetBucket = "amzn-s3-demo-bucket2";
    private const string SourceObjectKey = "example.mov";
    private const string TargetObjectKey = "copied_video_file.mov";

    /// <summary>
    /// This method starts the multi-part upload.
    /// </summary>
    public static async Task Main()
```

```
{
    var s3Client = new AmazonS3Client();
    Console.WriteLine("Copying object...");
    await MPUCopyObjectAsync(s3Client);
}

/// <summary>
/// This method uses the passed client object to perform a multipart
/// copy operation.
/// </summary>
/// <param name="client">An Amazon S3 client object that will be used
/// to perform the copy.</param>
public static async Task MPUCopyObjectAsync(AmazonS3Client client)
{
    // Create a list to store the copy part responses.
    var copyResponses = new List<CopyPartResponse>();

    // Setup information required to initiate the multipart upload.
    var initiateRequest = new InitiateMultipartUploadRequest
    {
        BucketName = TargetBucket,
        Key = TargetObjectKey,
    };

    // Initiate the upload.
    InitiateMultipartUploadResponse initResponse =
        await client.InitiateMultipartUploadAsync(initiateRequest);

    // Save the upload ID.
    string uploadId = initResponse.UploadId;

    try
    {
        // Get the size of the object.
        var metadataRequest = new GetObjectMetadataRequest
        {
            BucketName = SourceBucket,
            Key = SourceObjectKey,
        };

        GetObjectMetadataResponse metadataResponse =
            await client.GetObjectMetadataAsync(metadataRequest);
        var objectSize = metadataResponse.ContentLength; // Length in
bytes.
```

```
// Copy the parts.
var partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

long bytePosition = 0;
for (int i = 1; bytePosition < objectSize; i++)
{
    var copyRequest = new CopyPartRequest
    {
        DestinationBucket = TargetBucket,
        DestinationKey = TargetObjectKey,
        SourceBucket = SourceBucket,
        SourceKey = SourceObjectKey,
        UploadId = uploadId,
        FirstByte = bytePosition,
        LastByte = bytePosition + partSize - 1 >= objectSize ?
objectSize - 1 : bytePosition + partSize - 1,
        PartNumber = i,
    };

    copyResponses.Add(await client.CopyPartAsync(copyRequest));

    bytePosition += partSize;
}

// Set up to complete the copy.
var completeRequest = new CompleteMultipartUploadRequest
{
    BucketName = TargetBucket,
    Key = TargetObjectKey,
    UploadId = initResponse.UploadId,
};
completeRequest.AddPartETags(copyResponses);

// Complete the copy.
CompleteMultipartUploadResponse completeUploadResponse =
    await client.CompleteMultipartUploadAsync(completeRequest);
}
catch (AmazonS3Exception e)
{
    Console.WriteLine($"Error encountered on server.
Message: '{e.Message}' when writing an object");
}
catch (Exception e)
```

```
        {  
            Console.WriteLine($"Unknown encountered on server.  
Message: '{e.Message}' when writing an object");  
        }  
    }  
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per .NET .
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [GetObjectMetadata](#)
 - [UploadPartCopy](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Ricevi ed elabora le notifiche degli eventi di Amazon S3 utilizzando un AWS SDK

L'esempio di codice seguente mostra come utilizzare le notifiche di eventi S3 in modo orientato agli oggetti.

Java

SDK per Java 2.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Questo esempio mostra come elaborare un evento di notifica S3 utilizzando Amazon SQS.

```
/**  
 * This method receives S3 event notifications by using an SqsAsyncClient.
```

```

    * After the client receives the messages it deserializes the JSON payload
    and logs them. It uses
    * the S3EventNotification class (part of the S3 event notification API for
    Java) to deserialize
    * the JSON payload and access the messages in an object-oriented way.
    *
    * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
    notifications.
    * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
    awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification
    API</a>.
    * <p>
    * To use S3 event notification serialization/deserialization to objects, add
    the following
    * dependency to your Maven pom.xml file.
    * <dependency>
    * <groupId>software.amazon.awssdk</groupId>
    * <artifactId>s3-event-notifications</artifactId>
    * <version><LATEST></version>
    * </dependency>
    * <p>
    * The S3 event notification API became available with version 2.25.11 of the
    Java SDK.
    * <p>
    * This example shows the use of the API with AWS SQS, but it can be used to
    process S3 event notifications
    * in AWS SNS or AWS Lambda as well.
    * <p>
    * Note: The S3EventNotification class does not work with messages routed
    through AWS EventBridge.
    */
    static void processS3Events(String bucketName, String queueUrl, String
    queueArn) {
        try {
            // Configure the bucket to send Object Created and Object Tagging
            notifications to an existing SQS queue.
            s3Client.putBucketNotificationConfiguration(b -> b
                .notificationConfiguration(ncb -> ncb
                    .queueConfigurations(qcb -> qcb
                        .events(Event.S3_OBJECT_CREATED,
                            Event.S3_OBJECT_TAGGING)
                        .queueArn(queueArn)))
                    .bucket(bucketName)
                ).join();

```

```

        triggerS3EventNotifications(bucketName);
        // Wait for event notifications to propagate.
        Thread.sleep(Duration.ofSeconds(5).toMillis());

        boolean didReceiveMessages = true;
        while (didReceiveMessages) {
            // Display the number of messages that are available in the
queue.
            sqsClient.getQueueAttributes(b -> b
                .queueUrl(queueUrl)

                .attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
                    .thenAccept(attributeResponse ->
                        logger.info("Approximate number of messages in
the queue: {}"),
attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
                .join();

            // Receive the messages.
            ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
                .queueUrl(queueUrl)
                ).get();
            logger.info("Count of received messages: {}",
response.messages().size());
            didReceiveMessages = !response.messages().isEmpty();

            // Create a collection to hold the received message for deletion
            // after we log the messages.
            HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();

            // Process each message.
            response.messages().forEach(message -> {
                logger.info("Message id: {}", message.messageId());
                // Deserialize JSON message body to a S3EventNotification
object

                // to access messages in an object-oriented way.
                S3EventNotification event =
S3EventNotification.fromJson(message.body());

                // Log the S3 event notification record details.
                if (event.getRecords() != null) {
                    event.getRecords().forEach(record -> {

```



```

        String eventName = record.getEventName();
        String key = record.getS3().getObject().getKey();
        logger.info(record.toString());
        logger.info("Event name is {} and key is {}",
eventName, key);
    });
}
// Add logged messages to collection for batch deletion.
messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
    .id(message.messageId())
    .receiptHandle(message.receiptHandle())
    .build());
});
// Delete messages.
if (!messagesToDelete.isEmpty()) {

sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
    .queueUrl(queueUrl)
    .entries(messagesToDelete)
    .build()
    ).join();
}
} // End of while block.
} catch (InterruptedException | ExecutionException e) {
    throw new RuntimeException(e);
}
}
}

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK for Java 2.x .
 - [DeleteMessageBatch](#)
 - [GetQueueAttributes](#)
 - [PutBucketNotificationConfiguration](#)
 - [ReceiveMessage](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Salvate EXIF e altre informazioni sull'immagine utilizzando un AWS SDK

L'esempio di codice seguente mostra come:

- Recuperare informazioni EXIF da un file JPG, JPEG o PNG.
- Carica il file immagine in un bucket Amazon S3.
- Utilizza Amazon Rekognition per identificare i tre attributi principali (etichette) nel file.
- Aggiungi le informazioni su EXIF ed etichette a una tabella Amazon DynamoDB nella regione.

Rust

SDK per Rust

Recupera le informazioni EXIF da un file JPG, JPEG o PNG, carica il file di immagine in un bucket Amazon S3, utilizza Amazon Rekognition per identificare i tre attributi principali (etichette in Amazon Rekognition) nel file e aggiungi le informazioni su EXIF ed etichette a una tabella Amazon DynamoDB nella regione.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, guarda l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- DynamoDB
- Amazon Rekognition
- Simple Storage Service (Amazon S3)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Invia notifiche di eventi S3 ad Amazon EventBridge utilizzando un AWS SDK

Il seguente esempio di codice mostra come abilitare un bucket per inviare notifiche di eventi S3 EventBridge e indirizzarle verso un argomento Amazon SNS e una coda Amazon SQS.

Java

SDK per Java 2.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/** This method configures a bucket to send events to AWS EventBridge and
    creates a rule
    * to route the S3 object created events to a topic and a queue.
    *
    * @param bucketName Name of existing bucket
    * @param topicArn ARN of existing topic to receive S3 event notifications
    * @param queueArn ARN of existing queue to receive S3 event notifications
    *
    * An AWS CloudFormation stack sets up the bucket, queue, topic before the
    method runs.
    */
public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
    try {
        // Enable bucket to emit S3 Event notifications to EventBridge.
        s3Client.putBucketNotificationConfiguration(b -> b
            .bucket(bucketName)
            .notificationConfiguration(b1 -> b1
                .eventBridgeConfiguration(
                    SdkBuilder::build)
            ).build()).join();

        // Create an EventBridge rule to route Object Created notifications.
        PutRuleRequest putRuleRequest = PutRuleRequest.builder()
            .name(RULE_NAME)
            .eventPattern("""
                {
                    "source": ["aws.s3"],
                    "detail-type": ["Object Created"],
                    "detail": {
                        "bucket": {
                            "name": ["%s"]
                        }
                    }
                }
            """)
            .build();
        eventBridgeClient.putRule(putRuleRequest);
    } catch (Exception e) {
        // Handle exception
    }
}
```

```

        }
    }
}
"".formatted(bucketName))
    .build();

    // Add the rule to the default event bus.
    PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
    .whenComplete((r, t) -> {
        if (t != null) {
            logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                throw new RuntimeException(t.getCause().getMessage(),
t);
        }
        logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
    }).join();

    // Add the existing SNS topic and SQS queue as targets to the rule.
    eventBridgeClient.putTargets(b -> b
        .eventBusName("default")
        .rule(RULE_NAME)
        .targets(List.of (
            Target.builder()
                .arn(queueArn)
                .id("Queue")
                .build(),
            Target.builder()
                .arn(topicArn)
                .id("Topic")
                .build()
        )
    ).join();
    return putRuleResponse.ruleArn();
} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
}

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK for Java 2.x .
 - [PutBucketNotificationConfiguration](#)
 - [PutRule](#)
 - [PutTargets](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Tieni traccia del caricamento o del download di un oggetto Amazon S3 utilizzando un AWS SDK

L'esempio di codice seguente mostra come monitorare un caricamento o uno scaricamento di un oggetto Amazon S3.

Java

SDK per Java 2.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Monitora lo stato di avanzamento del caricamento di file.

```
public void trackUploadFile(S3TransferManager transferManager, String
bucketName,
                        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create()) // Add
listener.
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
```

```

        fileUpload.completionFuture().join();
    /*
        The SDK provides a LoggingTransferListener implementation of the
        TransferListener interface.
        You can also implement the interface to provide your own logic.

        Configure log4J2 with settings such as the following.
        <Configuration status="WARN">
            <Appenders>
                <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
                    <PatternLayout pattern="%m%n"/>
                </Console>
            </Appenders>

            <Loggers>
                <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                    <AppenderRef ref="AlignedConsoleAppender"/>
                </logger>
            </Loggers>
        </Configuration>

        Log4J2 logs the progress. The following is example output for a 21.3
        MB file upload.
        Transfer initiated...
        |                               | 0.0%
        |====                         | 21.1%
        |=====                      | 60.5%
        |=====                      | 100.0%
        Transfer complete!

    */
}

```

Monitora lo stato di avanzamento dello scaricamento di file.

```

public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                                String key, String downloadedFilePath) {
    DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()

```

```

        .getObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create()) // Add
listener.

        .destination(Paths.get(downloadedFilePath))
        .build();

```

```

FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

```

```

CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
/*

```

The SDK provides a `LoggingTransferListener` implementation of the `TransferListener` interface.

You can also implement the interface to provide your own logic.

Configure `log4j2` with settings such as the following.

```

<Configuration status="WARN">
  <Appenders>
    <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
      <PatternLayout pattern="%m%n"/>
    </Console>
  </Appenders>

  <Loggers>
    <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
      <AppenderRef ref="AlignedConsoleAppender"/>
    </logger>
  </Loggers>
</Configuration>

```

`Log4j2` logs the progress. The following is example output for a 21.3 MB file download.

```

Transfer initiated...
|=====| 39.4%
|=====| 78.8%
|=====| 100.0%
Transfer complete!

```

```

    */
}

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK for Java 2.x .
 - [GetObject](#)
 - [PutObject](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Trasformare i dati per l'applicazione con S3 Object Lambda

L'esempio di codice seguente mostra come trasformare i dati per l'applicazione con S3 Object Lambda.

.NET

SDK per .NET

Mostra come aggiungere codice personalizzato alle richieste S3 GET standard per modificare l'oggetto richiesto recuperato da S3 in modo che questo soddisfi le esigenze del client o dell'applicazione richiedente.

Per il codice sorgente completo e le istruzioni su come configurarlo ed eseguirlo, consulta l'esempio completo su [GitHub](#).

Servizi utilizzati in questo esempio

- Lambda
- Simple Storage Service (Amazon S3)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Esempi di approcci per test unitari e di integrazione con un AWS SDK

Il seguente esempio di codice mostra esempi di tecniche ottimali per la scrittura di test unitari e di integrazione utilizzando un AWS SDK.

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Cargo.toml per testare esempi.

```
[package]
name = "testing-examples"
version = "0.1.0"
authors = [
  "John Disanti <jdisanti@amazon.com>",
  "Doug Schwartz <dougsch@amazon.com>",
]
edition = "2021"

[dependencies]
async-trait = "0.1.51"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-credential-types = { version = "1.0.1", features = [ "hardcoded-credentials", ] }
aws-sdk-s3 = { version = "1.4.0" }
aws-smithy-types = { version = "1.0.1" }
aws-smithy-runtime = { version = "1.0.1", features = ["test-util"] }
aws-smithy-runtime-api = { version = "1.0.1", features = ["test-util"] }
aws-types = { version = "1.0.1" }
clap = { version = "4.4", features = ["derive"] }
http = "0.2.9"
mockall = "0.11.4"
serde_json = "1"
tokio = { version = "1.20.1", features = ["full"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }

[[bin]]
name = "main"
path = "src/main.rs"
```

Esempio di test di unità utilizzando automock e un wrapper di servizi.

```

use aws_sdk_s3 as s3;
#[allow(unused_imports)]
use mockall::automock;

use s3::operation::list_objects_v2::{ListObjectsV2Error, ListObjectsV2Output};

#[cfg(test)]
pub use MockS3Impl as S3;
#[cfg(not(test))]
pub use S3Impl as S3;

#[allow(dead_code)]
pub struct S3Impl {
    inner: s3::Client,
}

#[cfg_attr(test, automock)]
impl S3Impl {
    #[allow(dead_code)]
    pub fn new(inner: s3::Client) -> Self {
        Self { inner }
    }

    #[allow(dead_code)]
    pub async fn list_objects(
        &self,
        bucket: &str,
        prefix: &str,
        continuation_token: Option<String>,
    ) -> Result<ListObjectsV2Output, s3::error::SdkError<ListObjectsV2Error>> {
        self.inner
            .list_objects_v2()
            .bucket(bucket)
            .prefix(prefix)
            .set_continuation_token(continuation_token)
            .send()
            .await
    }
}

#[allow(dead_code)]

```

```

pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3_list: S3,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3_list
            .list_objects(bucket, prefix, next_token.take())
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}

#[cfg(test)]
mod test {
    use super::*;
    use mockall::predicate::eq;

    #[tokio::test]
    async fn test_single_page() {
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
            })
    }
}

```

```

        ]))
        .build())
    });

    // Run the code we want to test with it
    let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
        .await
        .unwrap();

    // Verify we got the correct total size back
    assert_eq!(7, size);
}

#[tokio::test]
async fn test_multiple_pages() {
    // Create the Mock instance with two pages of objects now
    let mut mock = MockS3Impl::default();
    mock.expect_list_objects()
        .with(eq("test-bucket"), eq("test-prefix"), eq(None))
        .return_once(|_, _, _| {
            Ok(ListObjectsV2Output::builder()
                .set_contents(Some(vec![
                    // Mock content for ListObjectsV2 response
                    s3::types::Object::builder().size(5).build(),
                    s3::types::Object::builder().size(2).build(),
                ]))
                .set_next_continuation_token(Some("next".to_string()))
                .build())
        });
    mock.expect_list_objects()
        .with(
            eq("test-bucket"),
            eq("test-prefix"),
            eq(Some("next".to_string())),
        )
        .return_once(|_, _, _| {
            Ok(ListObjectsV2Output::builder()
                .set_contents(Some(vec![
                    // Mock content for ListObjectsV2 response
                    s3::types::Object::builder().size(3).build(),
                    s3::types::Object::builder().size(9).build(),
                ]))
                .build())
        });
}

```

```

    // Run the code we want to test with it
    let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
        .await
        .unwrap();

    assert_eq!(19, size);
}
}

```

Esempio di test di integrazione utilizzando StaticReplayClient.

```

use aws_sdk_s3 as s3;

#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3: s3::Client,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3
            .list_objects_v2()
            .prefix(prefix)
            .bucket(bucket)
            .set_continuation_token(next_token.take())
            .send()
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {

```

```

        break;
    }
}
Ok(total_size_bytes)
}

#[allow(dead_code)]
fn make_s3_test_credentials() -> s3::config::Credentials {
    s3::config::Credentials::new(
        "ATESTCLIENT",
        "astestsecretkey",
        Some("atestsessiontoken".to_string()),
        None,
        "",
    )
}

#[cfg(test)]
mod test {
    use super::*;
    use aws_config::BehaviorVersion;
    use aws_sdk_s3 as s3;
    use aws_smithy_runtime::client::http::test_util::{ReplayEvent,
StaticReplayClient};
    use aws_smithy_types::body::SdkBody;

    #[tokio::test]
    async fn test_single_page() {
        let page_1 = ReplayEvent::new(
            http::Request::builder()
                .method("GET")
                .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                .body(SdkBody::empty())
                .unwrap(),
            http::Response::builder()
                .status(200)
                .body(SdkBody::from(include_str!("./testing/
response_1.xml")))
                .unwrap(),
        );
        let replay_client = StaticReplayClient::new(vec![page_1]);
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()

```

```

        .behavior_version(BehaviorVersion::latest())
        .credentials_provider(make_s3_test_credentials())
        .region(s3::config::Region::new("us-east-1"))
        .http_client(replay_client.clone())
        .build(),
    );

    // Run the code we want to test with it
    let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
        .await
        .unwrap();

    // Verify we got the correct total size back
    assert_eq!(7, size);
    replay_client.assert_requests_match(&[]);
}

#[tokio::test]
async fn test_multiple_pages() {
    let page_1 = ReplayEvent::new(
        http::Request::builder()
            .method("GET")
            .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
            .body(SdkBody::empty())
            .unwrap(),
        http::Response::builder()
            .status(200)
            .body(SdkBody::from(include_str!("./testing/
response_multi_1.xml")))
            .unwrap(),
    );
    let page_2 = ReplayEvent::new(
        http::Request::builder()
            .method("GET")
            .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix&continuation-token=next")
            .body(SdkBody::empty())
            .unwrap(),
        http::Response::builder()
            .status(200)
            .body(SdkBody::from(include_str!("./testing/
response_multi_2.xml")))

```

```
        .unwrap(),
    );
    let replay_client = StaticReplayClient::new(vec![page_1, page_2]);
    let client: s3::Client = s3::Client::from_conf(
        s3::Config::builder()
            .behavior_version(BehaviorVersion::latest())
            .credentials_provider(make_s3_test_credentials())
            .region(s3::config::Region::new("us-east-1"))
            .http_client(replay_client.clone())
            .build(),
    );

    // Run the code we want to test with it
    let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
        .await
        .unwrap();

    assert_eq!(19, size);

    replay_client.assert_requests_match(&[]);
}
}
```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Caricare in modo ricorsivo una directory locale in un bucket Amazon Simple Storage Service (Amazon S3)

L'esempio di codice seguente mostra come caricare una directory locale in modo ricorsivo in un bucket Amazon Simple Storage Service (Amazon S3).

Java

SDK per Java 2.x

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Usa un [S3TransferManager](#) per [caricare una directory locale](#). Visualizza il [file completo](#) ed esegui il [test](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;

    public Integer uploadDirectory(S3TransferManager transferManager,
        URI sourceDirectory, String bucketName) {
        DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
        .source(Paths.get(sourceDirectory))
        .bucket(bucketName)
        .build());

        CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers()
            .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

- Per i dettagli sull'API, consulta la [UploadDirectory](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Carica o scarica file di grandi dimensioni da e verso Amazon S3 utilizzando un AWS SDK

Gli esempi di codice seguente mostrano come caricare o scaricare file di grandi dimensioni in e da Amazon S3.

Per ulteriori informazioni, consulta [Caricamento di un oggetto utilizzando il caricamento in più parti](#).

.NET

SDK per .NET

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Chiama le funzioni che trasferiscono file da e verso un bucket S3 utilizzando Amazon S3.

TransferUtility

```
global using System.Text;
global using Amazon.S3;
global using Amazon.S3.Model;
global using Amazon.S3.Transfer;
global using TransferUtilityBasics;

// This Amazon S3 client uses the default user credentials
// defined for this computer.
using Microsoft.Extensions.Configuration;

IAmazonS3 client = new AmazonS3Client();
```

```
var transferUtil = new TransferUtility(client);
IConfiguration _configuration;

_configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from JSON file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

// Edit the values in settings.json to use an S3 bucket and files that
// exist on your AWS account and on the local computer where you
// run this scenario.
var bucketName = _configuration["BucketName"];
var localPath =
    $"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\
    \TransferFolder";

DisplayInstructions();

PressEnter();

Console.WriteLine();

// Upload a single file to an S3 bucket.
DisplayTitle("Upload a single file");

var fileToUpload = _configuration["FileToUpload"];
Console.WriteLine($"Uploading {fileToUpload} to the S3 bucket, {bucketName}.");

var success = await TransferMethods.UploadSingleFileAsync(transferUtil,
    bucketName, fileToUpload, localPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the file, {fileToUpload} to
    {bucketName}.");
}

PressEnter();

// Upload a local directory to an S3 bucket.
DisplayTitle("Upload all files from a local directory");
Console.WriteLine("Upload all the files in a local folder to an S3 bucket.");
const string keyPrefix = "UploadFolder";
```

```
var uploadPath = $"{localPath}\\UploadFolder";

Console.WriteLine($"Uploading the files in {uploadPath} to {bucketName}");
DisplayTitle($"{uploadPath} files");
DisplayLocalFiles(uploadPath);
Console.WriteLine();

PressEnter();

success = await TransferMethods.UploadFullDirectoryAsync(transferUtil,
    bucketName, keyPrefix, uploadPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the files in {uploadPath} to
{bucketName}.");
    Console.WriteLine($"{bucketName} currently contains the following files:");
    await DisplayBucketFiles(client, bucketName, keyPrefix);
    Console.WriteLine();
}

PressEnter();

// Download a single file from an S3 bucket.
DisplayTitle("Download a single file");
Console.WriteLine("Now we will download a single file from an S3 bucket.");

var keyName = _configuration["FileToDownload"];

Console.WriteLine($"Downloading {keyName} from {bucketName}.");

success = await TransferMethods.DownloadSingleFileAsync(transferUtil, bucketName,
    keyName, localPath);
if (success)
{
    Console.WriteLine($"Successfully downloaded the file, {keyName} from
{bucketName}.");
}

PressEnter();

// Download the contents of a directory from an S3 bucket.
DisplayTitle("Download the contents of an S3 bucket");
var s3Path = _configuration["S3Path"];
var downloadPath = $"{localPath}\\{s3Path}";
```

```
Console.WriteLine($"Downloading the contents of {bucketName}\\{s3Path}");
Console.WriteLine($"{bucketName}\\{s3Path} contains the following files:");
await DisplayBucketFiles(client, bucketName, s3Path);
Console.WriteLine();

success = await TransferMethods.DownloadS3DirectoryAsync(transferUtil,
    bucketName, s3Path, downloadPath);
if (success)
{
    Console.WriteLine($"Downloaded the files in {bucketName} to
{downloadPath}.");
    Console.WriteLine($"{downloadPath} now contains the following files:");
    DisplayLocalFiles(downloadPath);
}

Console.WriteLine("\nThe TransferUtility Basics application has completed.");
PressEnter();

// Displays the title for a section of the scenario.
static void DisplayTitle(string titleText)
{
    var sepBar = new string('-', Console.WindowWidth);

    Console.WriteLine(sepBar);
    Console.WriteLine(CenterText(titleText));
    Console.WriteLine(sepBar);
}

// Displays a description of the actions to be performed by the scenario.
static void DisplayInstructions()
{
    var sepBar = new string('-', Console.WindowWidth);

    DisplayTitle("Amazon S3 Transfer Utility Basics");
    Console.WriteLine("This program shows how to use the Amazon S3 Transfer
Utility.");
    Console.WriteLine("It performs the following actions:");
    Console.WriteLine("\t1. Upload a single object to an S3 bucket.");
    Console.WriteLine("\t2. Upload an entire directory from the local computer to
an\n\t S3 bucket.");
    Console.WriteLine("\t3. Download a single object from an S3 bucket.");
    Console.WriteLine("\t4. Download the objects in an S3 bucket to a local
directory.");
}
```

```
        Console.WriteLine($"\\n{sepBar}");
    }

    // Pauses the scenario.
    static void PressEnter()
    {
        Console.WriteLine("Press <Enter> to continue.");
        _ = Console.ReadLine();
        Console.WriteLine("\\n");
    }

    // Returns the string textToCenter, padded on the left with spaces
    // that center the text on the console display.
    static string CenterText(string textToCenter)
    {
        var centeredText = new StringBuilder();
        var screenWidth = Console.WindowWidth;
        centeredText.Append(new string(' ', (int)(screenWidth -
            textToCenter.Length) / 2));
        centeredText.Append(textToCenter);
        return centeredText.ToString();
    }

    // Displays a list of file names included in the specified path.
    static void DisplayLocalFiles(string localPath)
    {
        var fileList = Directory.GetFiles(localPath);
        if (fileList.Length > 0)
        {
            foreach (var fileName in fileList)
            {
                Console.WriteLine(fileName);
            }
        }
    }

    // Displays a list of the files in the specified S3 bucket and prefix.
    static async Task DisplayBucketFiles(IAmazonS3 client, string bucketName, string
        s3Path)
    {
        ListObjectsV2Request request = new()
        {
            BucketName = bucketName,
            Prefix = s3Path,
```

```

        MaxKeys = 5,
    };

    var response = new ListObjectsV2Response();

    do
    {
        response = await client.ListObjectsV2Async(request);

        response.S3Objects
            .ForEach(obj => Console.WriteLine($"{obj.Key}"));

        // If the response is truncated, set the request ContinuationToken
        // from the NextContinuationToken property of the response.
        request.ContinuationToken = response.NextContinuationToken;
    } while (response.IsTruncated);
}

```

Caricamento di un singolo file.

```

    /// <summary>
    /// Uploads a single file from the local computer to an S3 bucket.
    /// </summary>
    /// <param name="transferUtil">The transfer initialized TransferUtility
    /// object.</param>
    /// <param name="bucketName">The name of the S3 bucket where the file
    /// will be stored.</param>
    /// <param name="fileName">The name of the file to upload.</param>
    /// <param name="localPath">The local path where the file is stored.</
param>
    /// <returns>A boolean value indicating the success of the action.</
returns>
    public static async Task<bool> UploadSingleFileAsync(
        TransferUtility transferUtil,
        string bucketName,
        string fileName,
        string localPath)
    {
        if (File.Exists($"{localPath}\\{fileName}"))
        {

```

```

        try
        {
            await transferUtil.UploadAsync(new
TransferUtilityUploadRequest
            {
                BucketName = bucketName,
                Key = fileName,
                FilePath = $"{localPath}\\{fileName}",
            });

            return true;
        }
        catch (AmazonS3Exception s3Ex)
        {
            Console.WriteLine($"Could not upload {fileName} from
{localPath} because:");
            Console.WriteLine(s3Ex.Message);
            return false;
        }
    }
    else
    {
        Console.WriteLine($"{fileName} does not exist in {localPath}");
        return false;
    }
}

```

Caricamento di un'intera directory locale.

```

/// <summary>
/// Uploads all the files in a local directory to a directory in an S3
/// bucket.
/// </summary>
/// <param name="transferUtil">The transfer initialized TransferUtility
/// object.</param>
/// <param name="bucketName">The name of the S3 bucket where the files
/// will be stored.</param>
/// <param name="keyPrefix">The key prefix is the S3 directory where
/// the files will be stored.</param>
/// <param name="localPath">The local directory that contains the files
/// to be uploaded.</param>

```



```
/// <returns>A Boolean value representing the success of the action.</
returns>
public static async Task<bool> UploadFullDirectoryAsync(
    TransferUtility transferUtil,
    string bucketName,
    string keyPrefix,
    string localPath)
{
    if (Directory.Exists(localPath))
    {
        try
        {
            await transferUtil.UploadDirectoryAsync(new
TransferUtilityUploadDirectoryRequest
            {
                BucketName = bucketName,
                KeyPrefix = keyPrefix,
                Directory = localPath,
            });

            return true;
        }
        catch (AmazonS3Exception s3Ex)
        {
            Console.WriteLine($"Can't upload the contents of {localPath}
because:");
            Console.WriteLine(s3Ex?.Message);
            return false;
        }
    }
    else
    {
        Console.WriteLine($"The directory {localPath} does not exist.");
        return false;
    }
}
```

Download di un singolo file.

```
/// <summary>
```

```

    /// Download a single file from an S3 bucket to the local computer.
    /// </summary>
    /// <param name="transferUtil">The transfer initialized TransferUtility
    /// object.</param>
    /// <param name="bucketName">The name of the S3 bucket containing the
    /// file to download.</param>
    /// <param name="keyName">The name of the file to download.</param>
    /// <param name="localPath">The path on the local computer where the
    /// downloaded file will be saved.</param>
    /// <returns>A Boolean value indicating the results of the action.</
returns>
    public static async Task<bool> DownloadSingleFileAsync(
        TransferUtility transferUtil,
        string bucketName,
        string keyName,
        string localPath)
    {
        await transferUtil.DownloadAsync(new TransferUtilityDownloadRequest
        {
            BucketName = bucketName,
            Key = keyName,
            FilePath = $"{localPath}\\{keyName}",
        });

        return (File.Exists($"{localPath}\\{keyName}"));
    }

```

Download dei contenuti di un bucket S3.

```

    /// <summary>
    /// Downloads the contents of a directory in an S3 bucket to a
    /// directory on the local computer.
    /// </summary>
    /// <param name="transferUtil">The transfer initialized TransferUtility
    /// object.</param>
    /// <param name="bucketName">The bucket containing the files to
download.</param>
    /// <param name="s3Path">The S3 directory where the files are located.</
param>
    /// <param name="localPath">The local path to which the files will be

```

```
    /// saved.</param>
    /// <returns>A Boolean value representing the success of the action.</
returns>
    public static async Task<bool> DownloadS3DirectoryAsync(
        TransferUtility transferUtil,
        string bucketName,
        string s3Path,
        string localPath)
    {
        int fileCount = 0;

        // If the directory doesn't exist, it will be created.
        if (Directory.Exists(s3Path))
        {
            var files = Directory.GetFiles(localPath);
            fileCount = files.Length;
        }

        await transferUtil.DownloadDirectoryAsync(new
TransferUtilityDownloadDirectoryRequest
        {
            BucketName = bucketName,
            LocalDirectory = localPath,
            S3Directory = s3Path,
        });

        if (Directory.Exists(localPath))
        {
            var files = Directory.GetFiles(localPath);
            if (files.Length > fileCount)
            {
                return true;
            }

            // No change in the number of files. Assume
            // the download failed.
            return false;
        }

        // The local directory doesn't exist. No files
        // were downloaded.
        return false;
    }
}
```

Tieni traccia dello stato di avanzamento di un caricamento utilizzando `TransferUtility`

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Transfer;

/// <summary>
/// This example shows how to track the progress of a multipart upload
/// using the Amazon Simple Storage Service (Amazon S3) TransferUtility to
/// upload to an Amazon S3 bucket.
/// </summary>
public class TrackMPUUsingHighLevelAPI
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "sample_pic.png";
        string path = "filepath/directory/";
        string filePath = $"{path}{keyName}";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
        IAmazonS3 client = new AmazonS3Client();

        await TrackMPUAsync(client, bucketName, filePath, keyName);
    }

    /// <summary>
    /// Starts an Amazon S3 multipart upload and assigns an event handler to
    /// track the progress of the upload.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
    /// perform the multipart upload.</param>
    /// <param name="bucketName">The name of the bucket to which to upload
    /// the file.</param>
    /// <param name="filePath">The path, including the file name of the
    /// file to be uploaded to the Amazon S3 bucket.</param>
    /// <param name="keyName">The file name to be used in the
```

```

    /// destination Amazon S3 bucket.</param>
    public static async Task TrackMPUAsync(
        IAmazonS3 client,
        string bucketName,
        string filePath,
        string keyName)
    {
        try
        {
            var fileTransferUtility = new TransferUtility(client);

            // Use TransferUtilityUploadRequest to configure options.
            // In this example we subscribe to an event.
            var uploadRequest =
                new TransferUtilityUploadRequest
                {
                    BucketName = bucketName,
                    FilePath = filePath,
                    Key = keyName,
                };

            uploadRequest.UploadProgressEvent +=
                new EventHandler<UploadProgressArgs>(
                    UploadRequest_UploadPartProgressEvent);

            await fileTransferUtility.UploadAsync(uploadRequest);
            Console.WriteLine("Upload completed");
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error:: {ex.Message}");
        }
    }

    /// <summary>
    /// Event handler to check the progress of the multipart upload.
    /// </summary>
    /// <param name="sender">The object that raised the event.</param>
    /// <param name="e">The object that contains multipart upload
    /// information.</param>
    public static void UploadRequest_UploadPartProgressEvent(object sender,
        UploadProgressArgs e)
    {
        // Process event.
    }

```

```
        Console.WriteLine($"{e.TransferredBytes}/{e.TotalBytes}");  
    }  
}
```

Carica un oggetto con la crittografia.

```
using System;  
using System.Collections.Generic;  
using System.IO;  
using System.Security.Cryptography;  
using System.Threading.Tasks;  
using Amazon.S3;  
using Amazon.S3.Model;  
  
/// <summary>  
/// Uses the Amazon Simple Storage Service (Amazon S3) low level API to  
/// perform a multipart upload to an Amazon S3 bucket.  
/// </summary>  
public class SSECLowLevelMPUcopyObject  
{  
    public static async Task Main()  
    {  
        string existingBucketName = "amzn-s3-demo-bucket";  
        string sourceKeyName = "sample_file.txt";  
        string targetKeyName = "sample_file_copy.txt";  
        string filePath = $"sample\\{targetKeyName}";  
  
        // If the AWS Region defined for your default user is different  
        // from the Region where your Amazon S3 bucket is located,  
        // pass the Region name to the Amazon S3 client object's constructor.  
        // For example: RegionEndpoint.USEast1.  
        IAmazonS3 client = new AmazonS3Client();  
  
        // Create the encryption key.  
        var base64Key = CreateEncryptionKey();  
  
        await CreateSampleObjUsingClientEncryptionKeyAsync(  
            client,  
            existingBucketName,  
            sourceKeyName,  
            filePath,  
            base64Key);  
    }  
}
```

```

        base64Key);
    }

    /// <summary>
    /// Creates the encryption key to use with the multipart upload.
    /// </summary>
    /// <returns>A string containing the base64-encoded key for encrypting
    /// the multipart upload.</returns>
    public static string CreateEncryptionKey()
    {
        Aes aesEncryption = Aes.Create();
        aesEncryption.KeySize = 256;
        aesEncryption.GenerateKey();
        string base64Key = Convert.ToBase64String(aesEncryption.Key);
        return base64Key;
    }

    /// <summary>
    /// Creates and uploads an object using a multipart upload.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 object used to
    /// initialize and perform the multipart upload.</param>
    /// <param name="existingBucketName">The name of the bucket to which
    /// the object will be uploaded.</param>
    /// <param name="sourceKeyName">The source object name.</param>
    /// <param name="filePath">The location of the source object.</param>
    /// <param name="base64Key">The encryption key to use with the upload.</
param>
    public static async Task CreateSampleObjUsingClientEncryptionKeyAsync(
        IAmazonS3 client,
        string existingBucketName,
        string sourceKeyName,
        string filePath,
        string base64Key)
    {
        List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();

        InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
        {
            BucketName = existingBucketName,
            Key = sourceKeyName,

```

```
        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
        ServerSideEncryptionCustomerProvidedKey = base64Key,
    };

    InitiateMultipartUploadResponse initResponse =
        await client.InitiateMultipartUploadAsync(initiateRequest);

    long contentLength = new FileInfo(filePath).Length;
    long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

    try
    {
        long filePosition = 0;
        for (int i = 1; filePosition < contentLength; i++)
        {
            UploadPartRequest uploadRequest = new UploadPartRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                UploadId = initResponse.UploadId,
                PartNumber = i,
                PartSize = partSize,
                FilePosition = filePosition,
                FilePath = filePath,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            // Upload part and add response to our list.
            uploadResponses.Add(await
client.UploadPartAsync(uploadRequest));

            filePosition += partSize;
        }

        CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
        {
            BucketName = existingBucketName,
            Key = sourceKeyName,
            UploadId = initResponse.UploadId,
        };
    };
```



```

        completeRequest.AddPartETags(uploadResponses);

        CompleteMultipartUploadResponse completeUploadResponse =
            await client.CompleteMultipartUploadAsync(completeRequest);
    }
    catch (Exception exception)
    {
        Console.WriteLine($"Exception occurred: {exception.Message}");

        // If there was an error, abort the multipart upload.
        AbortMultipartUploadRequest abortMPURequest = new
        AbortMultipartUploadRequest
        {
            BucketName = existingBucketName,
            Key = sourceKeyName,
            UploadId = initResponse.UploadId,
        };

        await client.AbortMultipartUploadAsync(abortMPURequest);
    }
}

```

Go

SDK per Go V2

Note

C'è altro da fare GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea funzioni che utilizzano i gestori di caricamento e scaricamento per suddividere i dati in parti e trasferirli contemporaneamente.

```

import (
    "bytes"

```

```
"context"
"errors"
"fmt"
"io"
"log"
"os"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// UploadLargeObject uses an upload manager to upload data to an object in a
// bucket.
// The upload manager breaks large data into parts and uploads the parts
// concurrently.
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName
    string, objectKey string, largeObject []byte) error {
    largeBuffer := bytes.NewReader(largeObject)
    var partMiBs int64 = 10
    uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
        u.PartSize = partMiBs * 1024 * 1024
    })
    _, err := uploader.Upload(ctx, &s3.PutObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
        Body:    largeBuffer,
    })
    if err != nil {
```

```

var apiErr smithy.APIError
if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
    log.Printf("Error while uploading object to %s. The object is too large.\n"+
        "The maximum size for a multipart upload is 5TB.", bucketName)
} else {
    log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
        bucketName, objectKey, err)
}
} else {
    err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
        ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
aws.String(objectKey)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
    }
}

return err
}

// DownloadLargeObject uses a download manager to download an object from a
// bucket.
// The download manager gets the data in parts and writes them to a buffer until
// all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
string, objectKey string) ([]byte, error) {
    var partMiBs int64 = 10
    downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
    {
        d.PartSize = partMiBs * 1024 * 1024
    })
    buffer := manager.NewWriteAtBuffer([]byte{})
    _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:    aws.String(objectKey),
    })
    if err != nil {
        log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return buffer.Bytes(), err
}

```

```
}
```

Esegue uno scenario interattivo che mostra come utilizzare i gestori di caricamento e download nel contesto.

```
import (
    "context"
    "crypto/rand"
    "log"
    "strings"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunLargeObjectScenario is an interactive example that shows you how to use
// Amazon
// Simple Storage Service (Amazon S3) to upload and download large objects.
//
// 1. Create a bucket.
// 3. Upload a large object to the bucket by using an upload manager.
// 5. Download a large object by using a download manager.
// 8. Delete all objects in the bucket.
// 9. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
// so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../..demotools folder of this repo.
func RunLargeObjectScenario(ctx context.Context, sdkConfig aws.Config, questioner
    demotools.IQuestioner) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            _, isMock := questioner.(*demotools.MockQuestioner)
```

```
    if isMock || questioner.AskBool("Do you want to see the full error message (y/n)?", "y") {
        log.Println(r)
    }
}
}()

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon S3 large object demo.")
log.Println(strings.Repeat("-", 88))

s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}

bucketName := questioner.Ask("Let's create a bucket. Enter a name for your bucket:",
    demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
    panic(err)
}
if !bucketExists {
    err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
    if err != nil {
        panic(err)
    } else {
        log.Println("Bucket created.")
    }
}
log.Println(strings.Repeat("-", 88))

mibs := 30
log.Printf("Let's create a slice of %v MiB of random bytes and upload it to your bucket. ", mibs)
questioner.Ask("Press Enter when you're ready.")
largeBytes := make([]byte, 1024*1024*mibs)
_, _ = rand.Read(largeBytes)
largeKey := "doc-example-large"
log.Println("Uploading...")
err = bucketBasics.UploadLargeObject(ctx, bucketName, largeKey, largeBytes)
if err != nil {
    panic(err)
}
log.Printf("Uploaded %v MiB object as %v", mibs, largeKey)
```

```
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download the %v MiB object.", mibs)
questioner.Ask("Press Enter when you're ready.")
log.Println("Downloading...")
largeDownload, err := bucketBasics.DownloadLargeObject(ctx, bucketName,
largeKey)
if err != nil {
    panic(err)
}
log.Printf("Downloaded %v bytes.", len(largeDownload))
log.Println(strings.Repeat("-", 88))

if questioner.AskBool("Do you want to delete your bucket and all of its "+
"contents? (y/n)", "y") {
    log.Println("Deleting object.")
    err = bucketBasics.DeleteObjects(ctx, bucketName, []string{largeKey})
    if err != nil {
        panic(err)
    }
    log.Println("Deleting bucket.")
    err = bucketBasics.DeleteBucket(ctx, bucketName)
    if err != nil {
        panic(err)
    }
} else {
    log.Println("Okay. Don't forget to delete objects from your bucket to avoid
charges.")
}
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

Java

SDK per Java 2.x

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Richiama le funzioni che trasferiscono file da e verso un bucket S3 utilizzando.

S3TransferManager

```
public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
    .destination(Paths.get(destinationPathURI))
    .bucket(bucketName)
    .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}
```

Caricamento di un'intera directory locale.

```
public Integer uploadDirectory(S3TransferManager transferManager,
    URI sourceDirectory, String bucketName) {
    DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
    .source(Paths.get(sourceDirectory))
    .bucket(bucketName)
    .build());
    CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
}
```

```
        completedDirectoryUpload.failedTransfers()
            .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
}
```

Caricamento di un singolo file.

```
public String uploadFile(S3TransferManager transferManager, String
bucketName,
                        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

    CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
    return uploadResult.response().eTag();
}
```

Gli esempi di codice utilizzano le seguenti importazioni.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
```



```
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

Utilizza [S3 Transfer Manager](#) sulla parte superiore del [client AWS CRT-based S3](#) per eseguire in modo trasparente un caricamento in più parti quando la dimensione del contenuto supera una soglia. Le dimensioni soglia predefinite sono di 8 MB.

```
/**
 * Uploads a file to an Amazon S3 bucket using the S3TransferManager.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 * @param filePath    the file path of the file to be uploaded
 */
public void multipartUploadWithTransferManager(String bucketName, String key,
String filePath) {
    S3TransferManager transferManager = S3TransferManager.create();
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b
            .bucket(bucketName)
            .key(key))
        .source(Paths.get(filePath))
        .build();
    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
    fileUpload.completionFuture().join();
    transferManager.close();
}
```

Utilizza l'[API S3Client](#) per eseguire un caricamento in più parti.

```
/**
```

```
* Performs a multipart upload to Amazon S3 using the provided S3 client.
*
* @param bucketName the name of the S3 bucket
* @param key         the object key
* @param filePath    the path to the file to be uploaded
*/
public void multipartUploadWithS3Client(String bucketName, String key, String
filePath) {

    // Initiate the multipart upload.
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key));
    String uploadId = createMultipartUploadResponse.uploadId();

    // Upload the parts of the file.
    int partNumber = 1;
    List<CompletedPart> completedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

    try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
        long fileSize = file.length();
        long position = 0;
        while (position < fileSize) {
            file.seek(position);
            long read = file.getChannel().read(bb);

            bb.flip(); // Swap position and limit before reading from the
buffer.

            UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .partNumber(partNumber)
                .build();

            UploadPartResponse partResponse = s3Client.uploadPart(
                uploadPartRequest,
                RequestBody.fromByteBuffer(bb));

            CompletedPart part = CompletedPart.builder()
                .partNumber(partNumber)
                .eTag(partResponse.eTag())
```

```

        .build();
        completedParts.add(part);

        bb.clear();
        position += read;
        partNumber++;
    }
} catch (IOException e) {
    logger.error(e.getMessage());
}

// Complete the multipart upload.
s3Client.completeMultipartUpload(b -> b
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}

```

Utilizza l'[S3AsyncClient API](#) con il supporto multiparte abilitato per eseguire un caricamento in più parti.

```

/**
 * Uploads a file to an S3 bucket using the S3AsyncClient and enabling
 * multipart support.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 * @param filePath    the local file path of the file to be uploaded
 */
public void multipartUploadWithS3AsyncClient(String bucketName, String key,
String filePath) {
    // Enable multipart support.
    S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
        .multipartEnabled(true)
        .build();

    CompletableFuture<PutObjectResponse> response = s3AsyncClient.putObject(b
-> b
        .bucket(bucketName)

```

```
        .key(key),
        Paths.get(filePath));

    response.join();
    logger.info("File uploaded in multiple 8 MiB parts using
S3AsyncClient.");
}
```

JavaScript

SDK per (v3 JavaScript)

Note

C'è altro da fare. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Carica un file di grandi dimensioni.

```
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

import {
  ProgressBar,
  logger,
} from "@aws-doc-sdk-examples/lib/utils/util-log.js";

const twentyFiveMB = 25 * 1024 * 1024;

export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

/**
 * Create a 25MB file and upload it in parts to the specified
 * Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const str = createString();
```

```

const buffer = Buffer.from(str, "utf8");
const progressBar = new ProgressBar({
  description: `Uploading "${key}" to "${bucketName}"`,
  barLength: 30,
});

try {
  const upload = new Upload({
    client: new S3Client({}),
    params: {
      Bucket: bucketName,
      Key: key,
      Body: buffer,
    },
  });

  upload.on("httpUploadProgress", ({ loaded, total }) => {
    progressBar.update({ current: loaded, total });
  });

  await upload.done();
} catch (caught) {
  if (caught instanceof Error && caught.name === "AbortError") {
    logger.error(`Multipart upload was aborted. ${caught.message}`);
  } else {
    throw caught;
  }
}
};

```

Scarica un file di grandi dimensioni.

```

import { fileURLToPath } from "node:url";
import { GetObjectCommand, NoSuchKey, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream, rmSync } from "node:fs";

const s3Client = new S3Client({});
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
  const command = new GetObjectCommand({

```

```

    Bucket: bucket,
    Key: key,
    Range: `bytes=${start}-${end}`,
  });

  return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange
 */
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
  const [start, end] = range.split("-");
  return {
    start: Number.parseInt(start),
    end: Number.parseInt(end),
    length: Number.parseInt(length),
  };
};

export const isComplete = ({ end, length }) => end === length - 1;

const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));

  let rangeAndLength = { start: -1, end: -1, length: -1 };

  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
    const nextRange = { start: end + 1, end: end + oneMB };

    const { ContentRange, Body } = await getObjectRange({
      bucket,
      key,
      ...nextRange,
    });
    console.log(`Downloaded bytes ${nextRange.start} to ${nextRange.end}`);

    writeStream.write(await Body.transformToByteArray());
    rangeAndLength = getRangeAndLength(ContentRange);
  }
}

```

```
};

/**
 * Download a large object from and Amazon S3 bucket.
 *
 * When downloading a large file, you might want to break it down into
 * smaller pieces. Amazon S3 accepts a Range header to specify the start
 * and end of the byte range to be downloaded.
 *
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  try {
    await downloadInChunks({
      bucket: bucketName,
      key: key,
    });
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(`Failed to download object. No such key "${key}".`);
      rmSync(key);
    }
  }
};
```

Python

SDK per Python (Boto3)

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea funzioni che trasferiscono i file utilizzando diverse impostazioni disponibili del gestore di trasferimento. Utilizza una classe di callback per scrivere l'avanzamento del callback durante il trasferimento file.

```
import sys
```

```
import threading

import boto3
from boto3.s3.transfer import TransferConfig

MB = 1024 * 1024
s3 = boto3.resource("s3")

class TransferCallback:
    """
    Handle callbacks from the transfer manager.

    The transfer manager periodically calls the __call__ method throughout
    the upload and download process so that it can take action, such as
    displaying progress to the user and collecting data about the transfer.
    """

    def __init__(self, target_size):
        self._target_size = target_size
        self._total_transferred = 0
        self._lock = threading.Lock()
        self.thread_info = {}

    def __call__(self, bytes_transferred):
        """
        The callback method that is called by the transfer manager.

        Display progress during file transfer and collect per-thread transfer
        data. This method can be called by multiple threads, so shared instance
        data is protected by a thread lock.
        """
        thread = threading.current_thread()
        with self._lock:
            self._total_transferred += bytes_transferred
            if thread.ident not in self.thread_info.keys():
                self.thread_info[thread.ident] = bytes_transferred
            else:
                self.thread_info[thread.ident] += bytes_transferred

        target = self._target_size * MB
        sys.stdout.write(
            f"\r{self._total_transferred} of {target} transferred "
```



```

        f"({(self._total_transferred / target) * 100:.2f}%)."
    )
    sys.stdout.flush()

def upload_with_default_configuration(
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Callback=transfer_callback
    )
    return transfer_callback.thread_info

def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

    The multipart chunk size controls the size of the chunks of data that are
    sent in the request. A smaller chunk size typically results in the transfer
    manager using more threads for the upload.

    The metadata is a set of key-value pairs that are stored with the object
    in Amazon S3.
    """
    transfer_callback = TransferCallback(file_size_mb)

    config = TransferConfig(multipart_chunksize=1 * MB)
    extra_args = {"Metadata": metadata} if metadata else None
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Config=config,
        ExtraArgs=extra_args,
        Callback=transfer_callback,
    )

```

```
    return transfer_callback.thread_info

def upload_with_high_threshold(local_file_path, bucket_name, object_key,
    file_size_mb):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of
    a multipart upload.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info

def upload_with_sse(
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, ExtraArgs=extra_args,
Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

```
def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Callback=transfer_callback
    )
    return transfer_callback.thread_info

def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info

def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
```

```

        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info

def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
    )
    return transfer_callback.thread_info

```

Esegui le funzioni del gestore di trasferimento e ottieni i risultati.

```

import hashlib
import os
import platform
import shutil
import time

import boto3
from boto3.s3.transfer import TransferConfig
from botocore.exceptions import ClientError
from botocore.exceptions import ParamValidationError

```

```
from botocore.exceptions import NoCredentialsError

import file_transfer

MB = 1024 * 1024
# These configuration attributes affect both uploads and downloads.
CONFIG_ATTRS = (
    "multipart_threshold",
    "multipart_chunksize",
    "max_concurrency",
    "use_threads",
)
# These configuration attributes affect only downloads.
DOWNLOAD_CONFIG_ATTRS = ("max_io_queue", "io_chunksize", "num_download_attempts")

class TransferDemoManager:
    """
    Manages the demonstration. Collects user input from a command line, reports
    transfer results, maintains a list of artifacts created during the
    demonstration, and cleans them up after the demonstration is completed.
    """

    def __init__(self):
        self._s3 = boto3.resource("s3")
        self._chore_list = []
        self._create_file_cmd = None
        self._size_multiplier = 0
        self.file_size_mb = 30
        self.demo_folder = None
        self.demo_bucket = None
        self._setup_platform_specific()
        self._terminal_width = shutil.get_terminal_size(fallback=(80, 80))[0]

    def collect_user_info(self):
        """
        Collect local folder and Amazon S3 bucket name from the user. These
        locations are used to store files during the demonstration.
        """
        while not self.demo_folder:
            self.demo_folder = input(
                "Which file folder do you want to use to store " "demonstration
files? "
            )
```

```

        if not os.path.isdir(self.demo_folder):
            print(f"{self.demo_folder} isn't a folder!")
            self.demo_folder = None

    while not self.demo_bucket:
        self.demo_bucket = input(
            "Which Amazon S3 bucket do you want to use to store "
            "demonstration files? "
        )
    try:
        self._s3.meta.client.head_bucket(Bucket=self.demo_bucket)
    except ParamValidationError as err:
        print(err)
        self.demo_bucket = None
    except ClientError as err:
        print(err)
        print(
            f"Either {self.demo_bucket} doesn't exist or you don't "
            f"have access to it."
        )
        self.demo_bucket = None

    def demo(
        self, question, upload_func, download_func, upload_args=None,
download_args=None
    ):
        """Run a demonstration.

        Ask the user if they want to run this specific demonstration.
        If they say yes, create a file on the local path, upload it
        using the specified upload function, then download it using the
        specified download function.
        """
        if download_args is None:
            download_args = {}
        if upload_args is None:
            upload_args = {}
        question = question.format(self.file_size_mb)
        answer = input(f"{question} (y/n)")
        if answer.lower() == "y":
            local_file_path, object_key, download_file_path =
self._create_demo_file()

            file_transfer.TransferConfig = self._config_wrapper(

```

```

        TransferConfig, CONFIG_ATTRS
    )
    self._report_transfer_params(
        "Uploading", local_file_path, object_key, **upload_args
    )
    start_time = time.perf_counter()
    thread_info = upload_func(
        local_file_path,
        self.demo_bucket,
        object_key,
        self.file_size_mb,
        **upload_args,
    )
    end_time = time.perf_counter()
    self._report_transfer_result(thread_info, end_time - start_time)

    file_transfer.TransferConfig = self._config_wrapper(
        TransferConfig, CONFIG_ATTRS + DOWNLOAD_CONFIG_ATTRS
    )
    self._report_transfer_params(
        "Downloading", object_key, download_file_path, **download_args
    )
    start_time = time.perf_counter()
    thread_info = download_func(
        self.demo_bucket,
        object_key,
        download_file_path,
        self.file_size_mb,
        **download_args,
    )
    end_time = time.perf_counter()
    self._report_transfer_result(thread_info, end_time - start_time)

def last_name_set(self):
    """Get the name set used for the last demo."""
    return self._chore_list[-1]

def cleanup(self):
    """
    Remove files from the demo folder, and uploaded objects from the
    Amazon S3 bucket.
    """
    print("-" * self._terminal_width)

```

```

        for local_file_path, s3_object_key, downloaded_file_path in
self._chore_list:
    print(f"Removing {local_file_path}")
    try:
        os.remove(local_file_path)
    except FileNotFoundError as err:
        print(err)

    print(f"Removing {downloaded_file_path}")
    try:
        os.remove(downloaded_file_path)
    except FileNotFoundError as err:
        print(err)

    if self.demo_bucket:
        print(f"Removing {self.demo_bucket}:{s3_object_key}")
        try:

self._s3.Bucket(self.demo_bucket).Object(s3_object_key).delete()
            except ClientError as err:
                print(err)

def _setup_platform_specific(self):
    """Set up platform-specific command used to create a large file."""
    if platform.system() == "Windows":
        self._create_file_cmd = "fsutil file createnew {} {}".format(
            self._size_multiplier, MB
        )
    elif platform.system() == "Linux" or platform.system() == "Darwin":
        self._create_file_cmd = f"dd if=/dev/urandom of={{}} " f"bs={{MB}}
count={{}}"
        self._size_multiplier = 1
    else:
        raise EnvironmentError(
            f"Demo of platform {platform.system()} isn't supported."
        )

def _create_demo_file(self):
    """
    Create a file in the demo folder specified by the user. Store the local
    path, object name, and download path for later cleanup.

    Only the local file is created by this method. The Amazon S3 object and
    download file are created later during the demonstration.

```



```

Returns:
A tuple that contains the local file path, object name, and download
file path.
"""
file_name_template = "TestFile{}-{}.demo"
local_suffix = "local"
object_suffix = "s3object"
download_suffix = "downloaded"
file_tag = len(self._chore_list) + 1

local_file_path = os.path.join(
    self.demo_folder, file_name_template.format(file_tag, local_suffix)
)

s3_object_key = file_name_template.format(file_tag, object_suffix)

downloaded_file_path = os.path.join(
    self.demo_folder, file_name_template.format(file_tag,
download_suffix)
)

filled_cmd = self._create_file_cmd.format(
    local_file_path, self.file_size_mb * self._size_multiplier
)

print(
    f"Creating file of size {self.file_size_mb} MB "
    f"in {self.demo_folder} by running:"
)
print(f"{'':4}{filled_cmd}")
os.system(filled_cmd)

chore = (local_file_path, s3_object_key, downloaded_file_path)
self._chore_list.append(chore)
return chore

def _report_transfer_params(self, verb, source_name, dest_name, **kwargs):
    """Report configuration and extra arguments used for a file transfer."""
    print("-" * self._terminal_width)
    print(f"{verb} {source_name} ({self.file_size_mb} MB) to {dest_name}")
    if kwargs:
        print("With extra args:")
        for arg, value in kwargs.items():
            print(f"{'':4}{arg:<20}: {value}')

```

```

@staticmethod
def ask_user(question):
    """
    Ask the user a yes or no question.

    Returns:
    True when the user answers 'y' or 'Y'; otherwise, False.
    """
    answer = input(f"{question} (y/n) ")
    return answer.lower() == "y"

@staticmethod
def _config_wrapper(func, config_attrs):
    def wrapper(*args, **kwargs):
        config = func(*args, **kwargs)
        print("With configuration:")
        for attr in config_attrs:
            print(f'{"":4}{attr:<20}: {getattr(config, attr)}')
        return config

    return wrapper

@staticmethod
def _report_transfer_result(thread_info, elapsed):
    """Report the result of a transfer, including per-thread data."""
    print(f"\nUsed {len(thread_info)} threads.")
    for ident, byte_count in thread_info.items():
        print(f'{"":4}Thread {ident} copied {byte_count} bytes.')
    print(f"Your transfer took {elapsed:.2f} seconds.")

def main():
    """
    Run the demonstration script for s3_file_transfer.
    """
    demo_manager = TransferDemoManager()
    demo_manager.collect_user_info()

    # Upload and download with default configuration. Because the file is 30 MB
    # and the default multipart_threshold is 8 MB, both upload and download are
    # multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "

```

```
        "using the default configuration?",
        file_transfer.upload_with_default_configuration,
        file_transfer.download_with_default_configuration,
    )

    # Upload and download with multipart_threshold set higher than the size of
    # the file. This causes the transfer manager to use standard transfers
    # instead of multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "as a standard (not multipart) transfer?",
        file_transfer.upload_with_high_threshold,
        file_transfer.download_with_high_threshold,
    )

    # Upload with specific chunk size and additional metadata.
    # Download with a single thread.
    demo_manager.demo(
        "Do you want to upload a {} MB file with a smaller chunk size and "
        "then download the same file using a single thread?",
        file_transfer.upload_with_chunksize_and_meta,
        file_transfer.download_with_single_thread,
        upload_args={
            "metadata": {
                "upload_type": "chunky",
                "favorite_color": "aqua",
                "size": "medium",
            }
        },
    )

    # Upload using server-side encryption with customer-provided
    # encryption keys.
    # Generate a 256-bit key from a passphrase.
    sse_key = hashlib.sha256("demo_passphrase".encode("utf-8")).digest()
    demo_manager.demo(
        "Do you want to upload and download a {} MB file using "
        "server-side encryption?",
        file_transfer.upload_with_sse,
        file_transfer.download_with_sse,
        upload_args={"sse_key": sse_key},
        download_args={"sse_key": sse_key},
    )
```

```
# Download without specifying an encryption key to show that the
# encryption key must be included to download an encrypted object.
if demo_manager.ask_user(
    "Do you want to try to download the encrypted "
    "object without sending the required key?"
):
    try:
        _, object_key, download_file_path = demo_manager.last_name_set()
        file_transfer.download_with_default_configuration(
            demo_manager.demo_bucket,
            object_key,
            download_file_path,
            demo_manager.file_size_mb,
        )
    except ClientError as err:
        print(
            "Got expected error when trying to download an encrypted "
            "object without specifying encryption info:"
        )
        print(f"{'':4}{err}")

# Remove all created and downloaded files, remove all objects from
# S3 storage.
if demo_manager.ask_user(
    "Demonstration complete. Do you want to remove local files " "and S3
objects?"
):
    demo_manager.cleanup()

if __name__ == "__main__":
    try:
        main()
    except NoCredentialsError as error:
        print(error)
        print(
            "To run this example, you must have valid credentials in "
            "a shared credential file or set in environment variables."
        )
```

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
use std::fs::File;
use std::io::prelude::*;
use std::path::Path;

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::error::DisplayErrorContext;
use aws_sdk_s3::operation::{
    create_multipart_upload::CreateMultipartUploadOutput,
    get_object::GetObjectOutput,
};
use aws_sdk_s3::types::{CompletedMultipartUpload, CompletedPart};
use aws_sdk_s3::{config::Region, Client as S3Client};
use aws_smithy_types::byte_stream::{ByteStream, Length};
use rand::distributions::Alphanumeric;
use rand::{thread_rng, Rng};
use s3_code_examples::error::S3ExampleError;
use std::process;
use uuid::Uuid;

//In bytes, minimum chunk size of 5MB. Increase CHUNK_SIZE to send larger chunks.
const CHUNK_SIZE: u64 = 1024 * 1024 * 5;
const MAX_CHUNKS: u64 = 10000;

#[tokio::main]
pub async fn main() {
    if let Err(err) = run_example().await {
        eprintln!("Error: {}", DisplayErrorContext(err));
        process::exit(1);
    }
}
```

```
async fn run_example() -> Result<(), S3ExampleError> {
    let shared_config = aws_config::load_from_env().await;
    let client = S3Client::new(&shared_config);

    let bucket_name = format!("amzn-s3-demo-bucket-{}", Uuid::new_v4());
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();
    s3_code_examples::create_bucket(&client, &bucket_name, &region).await?;

    let key = "sample.txt".to_string();
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ))?;

    //Create a file of random characters for the upload.
    let mut file = File::create(&key).expect("Could not create sample file.");
    // Loop until the file is 5 chunks.
    while file.metadata().unwrap().len() <= CHUNK_SIZE * 4 {
        let rand_string: String = thread_rng()
            .sample_iter(&Alphanumeric)
            .take(256)
            .map(char::from)
            .collect();
        let return_string: String = "\n".to_string();
        file.write_all(rand_string.as_ref())
            .expect("Error writing to file.");
        file.write_all(return_string.as_ref())
            .expect("Error writing to file.");
    }

    let path = Path::new(&key);
    let file_size = tokio::fs::metadata(path)
        .await
        .expect("it exists I swear")
    }
```

```
.len());

let mut chunk_count = (file_size / CHUNK_SIZE) + 1;
let mut size_of_last_chunk = file_size % CHUNK_SIZE;
if size_of_last_chunk == 0 {
    size_of_last_chunk = CHUNK_SIZE;
    chunk_count -= 1;
}

if file_size == 0 {
    return Err(S3ExampleError::new("Bad file size.));
}
if chunk_count > MAX_CHUNKS {
    return Err(S3ExampleError::new(
        "Too many chunks! Try increasing your chunk size.",
    ));
}

let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
    let upload_part_res = client
        .upload_part()
        .key(&key)
        .bucket(&bucket_name)
        .upload_id(upload_id)
        .body(stream)
        .part_number(part_number)
        .send()
```

```

        .await?;

    upload_parts.push(
        CompletedPart::builder()
            .e_tag(upload_part_res.e_tag.unwrap_or_default())
            .part_number(part_number)
            .build(),
    );
}

// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;

let data: GetObjectOutput =
    s3_code_examples::download_object(&client, &bucket_name, &key).await?;
let data_length: u64 = data
    .content_length()
    .unwrap_or_default()
    .try_into()
    .unwrap();
if file.metadata().unwrap().len() == data_length {
    println!("Data lengths match.");
} else {
    println!("The data was not the same size!");
}

s3_code_examples::clear_bucket(&client, &bucket_name)
    .await
    .expect("Error emptying bucket.");
s3_code_examples::delete_bucket(&client, &bucket_name)
    .await
    .expect("Error deleting bucket.");

```



```
    ok(()  
}
```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Carica un flusso di dimensioni sconosciute su un oggetto Amazon S3 utilizzando un AWS SDK

Gli esempi di codice seguenti mostrano come caricare un flusso di dimensioni sconosciute in un oggetto Amazon S3.

Java

SDK per Java 2.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Usa il [client AWS CRT-based S3](#).

```
import com.example.s3.util.AsyncExampleUtils;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import software.amazon.awssdk.core.async.AsyncRequestBody;  
import software.amazon.awssdk.core.exception.SdkException;  
import software.amazon.awssdk.services.s3.S3AsyncClient;  
import software.amazon.awssdk.services.s3.model.PutObjectResponse;  
  
import java.io.ByteArrayInputStream;  
import java.io.InputStream;  
import java.util.UUID;  
import java.util.concurrent.CompletableFuture;  
import java.util.concurrent.ExecutorService;
```

```

import java.util.concurrent.Executors;

public class PutObjectFromStreamAsync {
    private static final Logger logger =
        LoggerFactory.getLogger(PutObjectFromStreamAsync.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.

        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            PutObjectFromStreamAsync example = new PutObjectFromStreamAsync();
            S3AsyncClient s3AsyncClientCrt = S3AsyncClient.crtCreate();
            PutObjectResponse putObjectResponse =
example.putObjectFromStreamCrt(s3AsyncClientCrt, bucketName, key);
            logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
            logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param s3CrtAsyncClient - To upload content from a stream of unknown
     size, use can the AWS CRT-based S3 client.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return software.amazon.awssdk.services.s3.model.PutObjectResponse -
     Returns metadata pertaining to the put object operation.
     */
    public PutObjectResponse putObjectFromStreamCrt(S3AsyncClient
s3CrtAsyncClient, String bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
        characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
    }
}

```

```

        InputStream inputStream = new
        ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
        thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
        length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
        null, executor);

        CompletableFuture<PutObjectResponse> responseFuture =
            s3CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
        body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
        response.
        logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
        executor.shutdown();
        return response;
    }
}

```

Utilizza il [client S3 asincrono standard con il caricamento in più parti abilitato](#).

```

import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class PutObjectFromStreamAsyncMp {

```

```

    private static final Logger logger =
        LoggerFactory.getLogger(PutObjectFromStreamAsyncMp.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.
        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            PutObjectFromStreamAsyncMp example = new
            PutObjectFromStreamAsyncMp();
            S3AsyncClient s3AsyncClientMp =
            S3AsyncClient.builder().multipartEnabled(true).build();
            PutObjectResponse putObjectResponse =
            example.putObjectFromStreamMp(s3AsyncClientMp, bucketName, key);
            logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
            logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param s3AsyncClientMp - To upload content from a stream of unknown size,
     use can the S3 asynchronous client with multipart enabled.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return software.amazon.awssdk.services.s3.model.PutObjectResponse -
     Returns metadata pertaining to the put object operation.
     */
    public PutObjectResponse putObjectFromStreamMp(S3AsyncClient s3AsyncClientMp,
        String bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
        characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
            randomString.length());
        InputStream inputStream = new
        ByteArrayInputStream(randomString.getBytes());

```

```
        // Executor required to handle reading from the InputStream on a separate
        thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
        length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
        null, executor);

        CompletableFuture<PutObjectResponse> responseFuture =
            s3AsyncClientMp.putObject(r -> r.bucket(bucketName).key(key),
        body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
        response.
        logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
        executor.shutdown();
        return response;
    }
}
```

Usa [Amazon S3 Transfer Manager](#).

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class UploadStream {
    private static final Logger logger =
        LoggerFactory.getLogger(UploadStream.class);
```

```
public static void main(String[] args) {
    String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
name.
    String key = UUID.randomUUID().toString();

    AsyncExampleUtils.createBucket(bucketName);
    try {
        UploadStream example = new UploadStream();
        CompletedUpload completedUpload =
example.uploadStream(S3TransferManager.create(), bucketName, key);
        logger.info("Object {} etag: {}", key,
completedUpload.response().eTag());
        logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
    } catch (SdkException e) {
        logger.error(e.getMessage(), e);
    } finally {
        AsyncExampleUtils.deleteObject(bucketName, key);
        AsyncExampleUtils.deleteBucket(bucketName);
    }
}

/**
 * @param transferManager - To upload content from a stream of unknown size,
you can use the S3TransferManager based on the AWS CRT-based S3 client.
 * @param bucketName - The name of the bucket.
 * @param key - The name of the object.
 * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
result of the completed upload.
 */
public CompletedUpload uploadStream(S3TransferManager transferManager, String
bucketName, String key) {

    // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
    String randomString = AsyncExampleUtils.randomString();
    logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
    InputStream inputStream = new
ByteArrayInputStream(randomString.getBytes());

    // Executor required to handle reading from the InputStream on a separate
thread so the main upload is not blocked.
    ExecutorService executor = Executors.newSingleThreadExecutor();
```

```
// Specify `null` for the content length when you don't know the content
length.
AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
null, executor);

Upload upload = transferManager.upload(builder -> builder
    .requestBody(body)
    .putObjectRequest(req -> req.bucket(bucketName).key(key))
    .build());

CompletedUpload completedUpload = upload.completionFuture().join();
executor.shutdown();
return completedUpload;
}
}
```

Swift

SDK per Swift

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import ArgumentParser
import AWSClientRuntime
import AWSS3
import Foundation
import Smithy
import SmithyHTTPAPI
import SmithyStreams

/// Upload a file to the specified bucket.
///
/// - Parameters:
///   - bucket: The Amazon S3 bucket name to store the file into.
///   - key: The name (or path) of the file to upload to in the `bucket`.
```

```
/// - sourcePath: The pathname on the local filesystem of the file to
///   upload.
func uploadFile(sourcePath: String, bucket: String, key: String?) async
throws {
    let fileURL: URL = URL(fileURLWithPath: sourcePath)
    let fileName: String

    // If no key was provided, use the last component of the filename.

    if key == nil {
        fileName = fileURL.lastPathComponent
    } else {
        fileName = key!
    }

    let s3Client = try await S3Client()

    // Create a FileHandle for the source file.

    let fileHandle = FileHandle(forReadingAtPath: sourcePath)
    guard let fileHandle = fileHandle else {
        throw TransferError.readError
    }

    // Create a byte stream to retrieve the file's contents. This uses the
    // Smithy FileStream and ByteStream types.

    let stream = FileStream(fileHandle: fileHandle)
    let body = ByteStream.stream(stream)

    // Create a `PutObjectInput` with the ByteStream as the body of the
    // request's data. The AWS SDK for Swift will handle sending the
    // entire file in chunks, regardless of its size.

    let putInput = PutObjectInput(
        body: body,
        bucket: bucket,
        key: fileName
    )

    do {
        _ = try await s3Client.putObject(input: putInput)
    } catch {
        throw TransferError.uploadError("Error uploading the file: \(error)")
    }
}
```



```
}  
  
    print("File uploaded to \"(fileURL.path).\"")  
}
```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Usa i checksum per lavorare con un oggetto Amazon S3 utilizzando un AWS SDK

L'esempio di codice seguente mostra come utilizzare i checksum per lavorare con un oggetto Amazon S3.

Java

SDK per Java 2.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Gli esempi di codice utilizzano un sottoinsieme delle seguenti importazioni.

```
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import software.amazon.awssdk.core.exception.SdkException;  
import software.amazon.awssdk.core.sync.RequestBody;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;  
import software.amazon.awssdk.services.s3.model.ChecksumMode;  
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;  
import software.amazon.awssdk.services.s3.model.CompletedPart;  
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;  
import software.amazon.awssdk.services.s3.model.GetObjectResponse;  
import software.amazon.awssdk.services.s3.model.UploadPartRequest;  
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
```

```
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

Specifica un algoritmo di checksum per il metodo `putObject` quando [crei il PutObjectRequest](#).

```
public void putObjectWithChecksum(String bucketName, String key) {
    s3Client.putObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(ChecksumAlgorithm.CRC32),
        RequestBody.fromString("This is a test"));
}
```

Verifica il checksum per il `getObject` metodo quando [crei il GetObjectRequest](#).

```
public GetObjectResponse getObjectWithChecksum(String bucketName, String key)
{
    return s3Client.getObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumMode(ChecksumMode.ENABLED))
        .response();
}
```

```
}
```

Pre-calcolate un checksum per il `putObject` metodo quando si [crea](#) il `PutObjectRequest`

```
public void putObjectWithPrecalculatedChecksum(String bucketName, String key,
String filePath) {
    String checksum = calculateChecksum(filePath, "SHA-256");

    s3Client.putObject((b -> b
        .bucket(bucketName)
        .key(key)
        .checksumSHA256(checksum)),
        RequestBody.fromFile(Paths.get(filePath)));
}
```

Utilizza [S3 Transfer Manager](#) sulla parte superiore del [client AWS CRT-based S3](#) per eseguire in modo trasparente un caricamento in più parti quando la dimensione del contenuto supera una soglia. Le dimensioni soglia predefinite sono di 8 MB.

Puoi specificare un algoritmo di checksum da utilizzare nell'SDK. Per impostazione predefinita, l'SDK utilizza l'algoritmo CRC32.

```
public void multipartUploadWithChecksumTm(String bucketName, String key,
String filePath) {
    S3TransferManager transferManager = S3TransferManager.create();
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b
            .bucket(bucketName)
            .key(key)
            .checksumAlgorithm(ChecksumAlgorithm.SHA1))
        .source(Paths.get(filePath))
        .build();
    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
    fileUpload.completionFuture().join();
    transferManager.close();
}
```

Utilizza l'[API o \(S3AsyncClient API\) S3Client](#) per eseguire un caricamento in più parti. Se specifichi un checksum aggiuntivo, devi specificare l'algoritmo da utilizzare all'avvio del

caricamento. Inoltre, devi specificare l'algoritmo per ogni richiesta parte e fornire il checksum calcolato per ciascuna parte dopo che è stata caricata.

```
public void multipartUploadWithChecksumS3Client(String bucketName, String
key, String filePath) {
    ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

    // Initiate the multipart upload.
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(algorithm)); // Checksum specified on initiation.
    String uploadId = createMultipartUploadResponse.uploadId();

    // Upload the parts of the file.
    int partNumber = 1;
    List<CompletedPart> completedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

    try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
        long fileSize = file.length();
        long position = 0;
        while (position < fileSize) {
            file.seek(position);
            long read = file.getChannel().read(bb);

            bb.flip(); // Swap position and limit before reading from the
buffer.

            UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .checksumAlgorithm(algorithm) // Checksum specified on each
part.

                .partNumber(partNumber)
                .build();

            UploadPartResponse partResponse = s3Client.uploadPart(
                uploadPartRequest,
                RequestBody.fromByteBuffer(bb));

            CompletedPart part = CompletedPart.builder()
```

```
        .partNumber(partNumber)
        .checksumCRC32(partResponse.checksumCRC32()) // Provide the
calculated checksum.
        .eTag(partResponse.eTag())
        .build();
    completedParts.add(part);

    bb.clear();
    position += read;
    partNumber++;
}
} catch (IOException e) {
    System.err.println(e.getMessage());
}

// Complete the multipart upload.
s3Client.completeMultipartUpload(b -> b
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)

    .multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK for Java 2.x .
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [UploadPart](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Lavora con le funzionalità di integrità degli oggetti di Amazon S3 utilizzando un AWS SDK

L'esempio di codice seguente mostra come utilizzare le funzionalità per l'integrità degli oggetti di S3.

C++

SDK per C++

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegue uno scenario interattivo che dimostra le funzionalità per l'integrità degli oggetti di Amazon S3.

```
#!/ Routine which runs the S3 object integrity workflow.
/*!
  \param clientConfig: Aws client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::S3::s3ObjectIntegrityWorkflow(
    const Aws::S3::S3ClientConfiguration &clientConfiguration) {

    /*
     * Create a large file to be used for multipart uploads.
     */
    if (!createLargeFileIfNotExists()) {
        std::cerr << "Workflow exiting because large file creation failed." <<
std::endl;
        return false;
    }

    Aws::String bucketName = TEST_BUCKET_PREFIX;
    bucketName += Aws::Utils::UUID::RandomUUID();
    bucketName = Aws::Utils::StringUtils::ToLower(bucketName.c_str());

    bucketName.resize(std::min(bucketName.size(), MAX_BUCKET_NAME_LENGTH));

    introductoryExplanations(bucketName);

    if (!AwsDoc::S3::createBucket(bucketName, clientConfiguration)) {
        std::cerr << "Workflow exiting because bucket creation failed." <<
std::endl;
        return false;
    }
}
```

```
}

Aws::S3::S3ClientConfiguration s3ClientConfiguration(clientConfiguration);
std::shared_ptr<Aws::S3::S3Client> client =
Aws::MakeShared<Aws::S3::S3Client>("S3Client", s3ClientConfiguration);

printAsterisksLine();
std::cout << "Choose from one of the following checksum algorithms."
          << std::endl;

for (HASH_METHOD hashMethod = DEFAULT; hashMethod <= SHA256; ++hashMethod) {
    std::cout << "  " << hashMethod << " - " <<
stringForHashMethod(hashMethod)
          << std::endl;
}

HASH_METHOD chosenHashMethod = askQuestionForIntRange("Enter an index: ",
DEFAULT,
                                                    SHA256);

gUseCalculatedChecksum = !askYesNoQuestion(
    "Let the SDK calculate the checksum for you? (y/n) ");

printAsterisksLine();

std::cout << "The workflow will now upload a file using PutObject."
          << std::endl;
std::cout << "Object integrity will be verified using the "
          << stringForHashMethod(chosenHashMethod) << " algorithm."
          << std::endl;
if (gUseCalculatedChecksum) {
    std::cout
        << "A checksum computed by this workflow will be used for object
integrity verification,"
        << std::endl;
    std::cout << "except for the TransferManager upload." << std::endl;
} else {
    std::cout
        << "A checksum computed by the SDK will be used for object
integrity verification."
        << std::endl;
}
```

```

pressEnterToContinue();
printAsterisksLine();

std::shared_ptr<Aws::IOStream> inputData =
    Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                  TEST_FILE,
                                  std::ios_base::in |
                                  std::ios_base::binary);

if (!*inputData) {
    std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}

Hasher hasher;
HASH_METHOD putObjectHashMethod = chosenHashMethod;
if (putObjectHashMethod == DEFAULT) {
    putObjectHashMethod = MD5; // MD5 is the default hash method for
PutObject.

    std::cout << "The default checksum algorithm for PutObject is "
               << stringForHashMethod(putObjectHashMethod)
               << std::endl;
}

// Demonstrate in code how the hash is computed.
if (!hasher.calculateObjectHash(*inputData, putObjectHashMethod)) {
    std::cerr << "Error calculating hash for file " << TEST_FILE <<
std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}
Aws::String key = stringForHashMethod(putObjectHashMethod);
key += "_";
key += TEST_FILE_KEY;
Aws::String localHash = hasher.getBase64HashString();

// Upload the object with PutObject
if (!putObjectWithHash(bucketName, key, localHash, putObjectHashMethod,
                      inputData, chosenHashMethod == DEFAULT,
                      *client)) {
    std::cerr << "Error putting file " << TEST_FILE << " to bucket "
               << bucketName << " with key " << key << std::endl;
}

```



```
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    Aws::String retrievedHash;
    if (!retrieveObjectHash(bucketName, key,
                           putObjectHashMethod, retrievedHash,
                           nullptr, *client)) {
        std::cerr << "Error getting file " << TEST_FILE << " from bucket "
                   << bucketName << " with key " << key << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    explainPutObjectResults();
    verifyHashingResults(retrievedHash, hasher,
                        "PutObject upload", putObjectHashMethod);

    printAsterisksLine();
    pressEnterToContinue();

    key = "tr_";
    key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

    introductoryTransferManagerUploadExplanations(key);

    HASH_METHOD transferManagerHashMethod = chosenHashMethod;
    if (transferManagerHashMethod == DEFAULT) {
        transferManagerHashMethod = CRC32; // The default hash method for the
        TransferManager is CRC32.

        std::cout << "The default checksum algorithm for TransferManager is "
                  << stringForHashMethod(transferManagerHashMethod)
                  << std::endl;
    }

    // Upload the large file using the transfer manager.
    if (!doTransferManagerUpload(bucketName, key, transferManagerHashMethod,
                                chosenHashMethod == DEFAULT,
                                client)) {
        std::cerr << "Exiting because of an error in doTransferManagerUpload." <<
        std::endl;
        cleanUp(bucketName, clientConfiguration);
    }
}
```

```

        return false;
    }

    std::vector<Aws::String> retrievedTransferManagerPartHashes;
    Aws::String retrievedTransferManagerFinalHash;

    // Retrieve all the hashes for the TransferManager upload.
    if (!retrieveObjectHash(bucketName, key,
                           transferManagerHashMethod,
                           retrievedTransferManagerFinalHash,
                           &retrievedTransferManagerPartHashes, *client)) {
        std::cerr << "Exiting because of an error in retrieveObjectHash for
TransferManager." << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    AwsDoc::S3::Hasher locallyCalculatedFinalHash;
    std::vector<Aws::String> locallyCalculatedPartHashes;

    // Calculate the hashes locally to demonstrate how TransferManager hashes are
    computed.
    if (!calculatePartHashesForFile(transferManagerHashMethod,
MULTI_PART_TEST_FILE,
                                   UPLOAD_BUFFER_SIZE,
                                   locallyCalculatedFinalHash,
                                   locallyCalculatedPartHashes)) {
        std::cerr << "Exiting because of an error in calculatePartHashesForFile."
<< std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    verifyHashingResults(retrievedTransferManagerFinalHash,
                        locallyCalculatedFinalHash, "TransferManager upload",
                        transferManagerHashMethod,
                        retrievedTransferManagerPartHashes,
                        locallyCalculatedPartHashes);

    printAsterisksLine();

    key = "mp_";
    key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

```

```
multiPartUploadExplanations(key, chosenHashMethod);

pressEnterToContinue();

std::shared_ptr<Aws::IOStream> largeFileInputData =
    Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                  MULTI_PART_TEST_FILE,
                                  std::ios_base::in |
                                  std::ios_base::binary);

if (!largeFileInputData->good()) {
    std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}

HASH_METHOD multipartUploadHashMethod = chosenHashMethod;
if (multipartUploadHashMethod == DEFAULT) {
    multipartUploadHashMethod = MD5; // The default hash method for
multipart uploads is MD5.

    std::cout << "The default checksum algorithm for multipart upload is "
                << stringForHashMethod(putObjectHashMethod)
                << std::endl;
}

AwsDoc::S3::Hasher hashData;
std::vector<Aws::String> partHashes;

if (!doMultipartUpload(bucketName, key,
                        multipartUploadHashMethod,
                        largeFileInputData, chosenHashMethod == DEFAULT,
                        hashData,
                        partHashes,
                        *client)) {
    std::cerr << "Exiting because of an error in doMultipartUpload." <<
std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}

std::cout << "Finished multipart upload of with hash method " <<
            stringForHashMethod(multipartUploadHashMethod) << std::endl;
```

```

    std::cout << "Now we will retrieve the checksums from the server." <<
    std::endl;

    retrievedHash.clear();
    std::vector<Aws::String> retrievedPartHashes;
    if (!retrieveObjectHash(bucketName, key,
                           multipartUploadHashMethod,
                           retrievedHash, &retrievedPartHashes, *client)) {
        std::cerr << "Exiting because of an error in retrieveObjectHash for
multipart." << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    verifyHashingResults(retrievedHash, hashData, "MultiPart upload",
                        multipartUploadHashMethod,
                        retrievedPartHashes, partHashes);

    printAsterisksLine();

    if (askYesNoQuestion("Would you like to delete the resources created in this
workflow? (y/n)")) {
        return cleanUp(bucketName, clientConfiguration);
    } else {
        std::cout << "The bucket " << bucketName << " was not deleted." <<
        std::endl;
        return true;
    }
}

//! Routine which uploads an object to an S3 bucket with different object
integrity hashing methods.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashData: The hash value that will be associated with the uploaded
object.
    \param hashMethod: The hashing algorithm to use when calculating the hash
value.
    \param body: The data content of the object being uploaded.
    \param useDefaultHashMethod: A flag indicating whether to use the default hash
method or the one specified in the hashMethod parameter.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.

```

```

*/
bool AwsDoc::S3::putObjectWithHash(const Aws::String &bucket, const Aws::String
    &key,
                                const Aws::String &hashData,
                                AwsDoc::S3::HASH_METHOD hashMethod,
                                const std::shared_ptr<Aws::IOStream> &body,
                                bool useDefaultHashMethod,
                                const Aws::S3::S3Client &client) {
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {
            request.SetChecksumAlgorithm(getChecksumAlgorithmForHashMethod(hashMethod));
        }
    }

    if (gUseCalculatedChecksum) {
        switch (hashMethod) {
            case AwsDoc::S3::MD5:
                request.SetContentMD5(hashData);
                break;
            case AwsDoc::S3::SHA1:
                request.SetChecksumSHA1(hashData);
                break;
            case AwsDoc::S3::SHA256:
                request.SetChecksumSHA256(hashData);
                break;
            case AwsDoc::S3::CRC32:
                request.SetChecksumCRC32(hashData);
                break;
            case AwsDoc::S3::CRC32C:
                request.SetChecksumCRC32C(hashData);
                break;
            default:
                std::cerr << "Unknown hash method." << std::endl;
                return false;
        }
    }
    request.SetBody(body);
    Aws::S3::Model::PutObjectOutcome outcome = client.PutObject(request);
    body->seekg(0, body->beg);
    if (outcome.IsSuccess()) {

```

```

        std::cout << "Object successfully uploaded." << std::endl;
    } else {
        std::cerr << "Error uploading object." <<
            outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}

// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object is stored.
    \param key: The unique identifier (key) of the object within the S3 bucket.
    \param hashMethod: The hashing algorithm used to calculate the hash value of
the object.
    \param[out] hashData: The retrieved hash.
    \param[out] partHashes: The part hashes if available.
    \param client: The S3 client instance used to retrieve the object.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
&key,

                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   Aws::String &hashData,
                                   std::vector<Aws::String> *partHashes,
                                   const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
            hashData = result.GetETag();
        } else {
            std::cerr << "Error retrieving object etag attributes." <<

```

```

        outcome.GetError().GetMessage() << std::endl;
        return false;
    }
} else { // hashMethod != MD5
    Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
    attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
    request.SetObjectAttributes(attributes);

    Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        switch (hashMethod) {
            case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                break; // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
            case AwsDoc::S3::MD5:
                break; // MD5 is not supported.
#pragma clang diagnostic pop
            case AwsDoc::S3::SHA1:
                hashData = result.GetChecksum().GetChecksumSHA1();
                break;
            case AwsDoc::S3::SHA256:
                hashData = result.GetChecksum().GetChecksumSHA256();
                break;
            case AwsDoc::S3::CRC32:
                hashData = result.GetChecksum().GetChecksumCRC32();
                break;
            case AwsDoc::S3::CRC32C:
                hashData = result.GetChecksum().GetChecksumCRC32C();
                break;
            default:
                std::cerr << "Unknown hash method." << std::endl;
                return false;
        }
    } else {
        std::cerr << "Error retrieving object checksum attributes." <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

```

```

        if (nullptr != partHashes) {
            attributes.clear();
            attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
            request.SetObjectAttributes(attributes);
            outcome = client.GetObjectAttributes(request);
            if (outcome.IsSuccess()) {
                const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
                const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
                for (const Aws::S3::Model::ObjectPart &part: parts) {
                    switch (hashMethod) {
                        case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                            break;
                        case AwsDoc::S3::MD5: // MD5 is not supported.
                            break;
                        case AwsDoc::S3::SHA1:
                            partHashes->push_back(part.GetChecksumSHA1());
                            break;
                        case AwsDoc::S3::SHA256:
                            partHashes->push_back(part.GetChecksumSHA256());
                            break;
                        case AwsDoc::S3::CRC32:
                            partHashes->push_back(part.GetChecksumCRC32());
                            break;
                        case AwsDoc::S3::CRC32C:
                            partHashes->push_back(part.GetChecksumCRC32C());
                            break;
                        default:
                            std::cerr << "Unknown hash method." << std::endl;
                            return false;
                    }
                }
            } else {
                std::cerr << "Error retrieving object attributes for object
parts." <<
                    outcome.GetError().GetMessage() << std::endl;
                return false;
            }
        }
    }
}

return true;

```



```

}

//! Verifies the hashing results between the retrieved and local hashes.
/*!
\param retrievedHash The hash value retrieved from the remote source.
\param localHash The hash value calculated locally.
\param uploadtype The type of upload (e.g., "multipart", "single-part").
\param hashMethod The hashing method used (e.g., MD5, SHA-256).
\param retrievedPartHashes (Optional) The list of hashes for the individual
parts retrieved from the remote source.
\param localPartHashes (Optional) The list of hashes for the individual parts
calculated locally.
*/
void AwsDoc::S3::verifyHashingResults(const Aws::String &retrievedHash,
                                     const Hasher &localHash,
                                     const Aws::String &uploadtype,
                                     HASH_METHOD hashMethod,
                                     const std::vector<Aws::String>
&retrievedPartHashes,
                                     const std::vector<Aws::String>
&localPartHashes) {
    std::cout << "For " << uploadtype << " retrieved hash is " << retrievedHash
<< std::endl;
    if (!retrievedPartHashes.empty()) {
        std::cout << retrievedPartHashes.size() << " part hash(es) were also
retrieved."
<< std::endl;
        for (auto &retrievedPartHash: retrievedPartHashes) {
            std::cout << " Part hash " << retrievedPartHash << std::endl;
        }
    }
    Aws::String hashString;
    if (hashMethod == MD5) {
        hashString = localHash.getHexHashString();
        if (!localPartHashes.empty()) {
            hashString += "-" + std::to_string(localPartHashes.size());
        }
    } else {
        hashString = localHash.getBase64HashString();
    }

    bool allMatch = true;
    if (hashString != retrievedHash) {

```

```

        std::cerr << "For " << uploadtype << ", the main hashes do not match" <<
std::endl;
        std::cerr << "Local hash- '" << hashString << "'" << std::endl;
        std::cerr << "Remote hash - '" << retrievedHash << "'" << std::endl;
        allMatch = false;
    }

    if (hashMethod != MD5) {
        if (localPartHashes.size() != retrievedPartHashes.size()) {
            std::cerr << "For " << uploadtype << ", the number of part hashes do
not match" << std::endl;
            std::cerr << "Local number of hashes- '" << localPartHashes.size() <<
""
                << std::endl;
            std::cerr << "Remote number of hashes - '"
                << retrievedPartHashes.size()
                << "'" << std::endl;
        }

        for (int i = 0; i < localPartHashes.size(); ++i) {
            if (localPartHashes[i] != retrievedPartHashes[i]) {
                std::cerr << "For " << uploadtype << ", the part hashes do not
match for part " << i + 1
                    << "." << std::endl;
                std::cerr << "Local hash- '" << localPartHashes[i] << "'"
                    << std::endl;
                std::cerr << "Remote hash - '" << retrievedPartHashes[i] << "'"
                    << std::endl;
                allMatch = false;
            }
        }
    }

    if (allMatch) {
        std::cout << "For " << uploadtype << ", locally and remotely calculated
hashes all match!" << std::endl;
    }
}

static void transferManagerErrorCallback(const Aws::Transfer::TransferManager *,
                                         const std::shared_ptr<const
    Aws::Transfer::TransferHandle> &,

```

```

                                const
    Aws::Client::AWSError<Aws::S3::S3Errors> &err) {
        std::cerr << "Error during transfer: " << err.GetMessage() << "" <<
        std::endl;
    }

    static void transferManagerStatusCallback(const Aws::Transfer::TransferManager *,
                                              const std::shared_ptr<const
    Aws::Transfer::TransferHandle> &handle) {
        if (handle->GetStatus() == Aws::Transfer::TransferStatus::IN_PROGRESS) {
            std::cout << "Bytes transferred: " << handle->GetBytesTransferred() <<
            std::endl;
        }
    }
}

//! Routine which uploads an object to an S3 bucket using the AWS C++ SDK's
    Transfer Manager.
    /*!
        \param bucket: The name of the S3 bucket where the object will be uploaded.
        \param key: The unique identifier (key) for the object within the S3 bucket.
        \param hashMethod: The hashing algorithm to use when calculating the hash
        value.
        \param useDefaultHashMethod: A flag indicating whether to use the default hash
        method or the one specified in the hashMethod parameter.
        \param client: The S3 client instance used to perform the upload operation.
        \return bool: Function succeeded.
    */
    bool
    AwsDoc::S3::doTransferManagerUpload(const Aws::String &bucket, const Aws::String
    &key,

                                AwsDoc::S3::HASH_METHOD hashMethod,
                                bool useDefaultHashMethod,
                                const std::shared_ptr<Aws::S3::S3Client>

    &client) {
        std::shared_ptr<Aws::Utils::Threading::PooledThreadExecutor> executor =
    Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
        "executor", 25);
        Aws::Transfer::TransferManagerConfiguration transfer_config(executor.get());
        transfer_config.s3Client = client;
        transfer_config.bufferSize = UPLOAD_BUFFER_SIZE;
        if (!useDefaultHashMethod) {
            if (hashMethod == MD5) {
                transfer_config.computeContentMD5 = true;
            } else {

```

```

        transfer_config.checksumAlgorithm =
getChecksumAlgorithmForHashMethod(
            hashMethod);
    }
}
transfer_config.errorCallback = transferManagerErrorCallback;
transfer_config.transferStatusUpdatedCallback =
transferManagerStatusCallback;

    std::shared_ptr<Aws::Transfer::TransferManager> transfer_manager =
Aws::Transfer::TransferManager::Create(
    transfer_config);

    std::cout << "Uploading the file..." << std::endl;
    std::shared_ptr<Aws::Transfer::TransferHandle> uploadHandle =
transfer_manager->UploadFile(MULTI_PART_TEST_FILE,

        bucket, key,

        "text/plain",

        Aws::Map<Aws::String, Aws::String>());
    uploadHandle->WaitUntilFinished();
    bool success =
        uploadHandle->GetStatus() ==
Aws::Transfer::TransferStatus::COMPLETED;
    if (!success) {
        Aws::Client::AWSError<Aws::S3::S3Errors> err = uploadHandle-
>GetLastError();
        std::cerr << "File upload failed:  " << err.GetMessage() << std::endl;
    }

    return success;
}

//! Routine which calculates the hash values for each part of a file being
    uploaded to an S3 bucket.
/*!
    \param hashMethod: The hashing algorithm to use when calculating the hash
    values.
    \param fileName: The path to the file for which the part hashes will be
    calculated.
    \param bufferSize: The size of the buffer to use when reading the file.

```

```

    \param[out] hashDataResult: The Hasher object that will store the concatenated
    hash value.
    \param[out] partHashes: The vector that will store the calculated hash values
    for each part of the file.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::calculatePartHashesForFile(AwsDoc::S3::HASH_METHOD hashMethod,
                                             const Aws::String &fileName,
                                             size_t bufferSize,
                                             AwsDoc::S3::Hasher &hashDataResult,
                                             std::vector<Aws::String> &partHashes)
{
    std::ifstream fileStream(fileName.c_str(), std::ifstream::binary);
    fileStream.seekg(0, std::ifstream::end);
    size_t objectSize = fileStream.tellg();
    fileStream.seekg(0, std::ifstream::beg);
    std::vector<unsigned char> totalHashBuffer;
    size_t uploadedBytes = 0;

    while (uploadedBytes < objectSize) {
        std::vector<unsigned char> buffer(bufferSize);
        std::streamsize bytesToRead =
static_cast<std::streamsize>(std::min(buffer.size(), objectSize -
uploadedBytes));
        fileStream.read((char *) buffer.data(), bytesToRead);
        Aws::Utils::Stream::PreallocatedStreamBuf
preallocatedStreamBuf(buffer.data(),

bytesToRead);
        std::shared_ptr<Aws::IOStream> body =
            Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                                           &preallocatedStreamBuf);

        Hasher hasher;
        if (!hasher.calculateObjectHash(*body, hashMethod)) {
            std::cerr << "Error calculating hash." << std::endl;
            return false;
        }
        Aws::String base64HashString = hasher.getBase64HashString();
        partHashes.push_back(base64HashString);

        Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();
    }
}

```

```

        totalHashBuffer.insert(totalHashBuffer.end(),
hashBuffer.GetUnderlyingData(),
                                hashBuffer.GetUnderlyingData() +
hashBuffer.GetLength());

        uploadedBytes += bytesToRead;
    }

    return hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod);
}

//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String
&key,
                                Aws::S3::Model::ChecksumAlgorithm
checksumAlgorithm,
                                const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
        client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }

    return uploadID;
}

```

```

}

//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the
checksum algorithm,
                                ignored when it is an empty string.
    \param body: An shared_ptr IOSTream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String
&bucket,
                                                         const Aws::String &key,
                                                         const Aws::String
&uploadID,
                                                         int partNumber,
                                                         Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                         const Aws::String
&calculatedHash,
                                                         const
std::shared_ptr<Aws::IOStream> &body,
                                                         const Aws::S3::S3Client
&client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:

```

```

        request.SetContentMD5(calculatedHash);
        break;
    case Aws::S3::Model::ChecksumAlgorithm::CRC32:
        request.SetChecksumCRC32(calculatedHash);
        break;
    case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
        request.SetChecksumCRC32C(calculatedHash);
        break;
    case Aws::S3::Model::ChecksumAlgorithm::SHA1:
        request.SetChecksumSHA1(calculatedHash);
        break;
    case Aws::S3::Model::ChecksumAlgorithm::SHA256:
        request.SetChecksumSHA256(calculatedHash);
        break;
    }
}

return client.UploadPart(request);
}

//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                       const Aws::String &key,
                                       const Aws::String &uploadID,
                                       const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
        client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {

```



```

        std::cerr << "Error aborting multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

///  

// Complete a multipart upload to an S3 bucket.  

//  

// \param bucket: The name of the S3 bucket where the object will be uploaded.  

// \param key: The unique identifier (key) for the object within the S3 bucket.  

// \param uploadID: An upload ID string.  

// \param parts: A vector of CompleteParts.  

// \param client: The S3 client instance used to perform the upload operation.  

// \return CompleteMultipartUploadOutcome: The request outcome.  

//  

Aws::S3::Model::CompleteMultipartUploadOutcome  

AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,  

    const Aws::String &key,  

    const Aws::String &uploadID,  

    const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,  

    const Aws::S3::S3Client &client) {  

    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;  

    completedMultipartUpload.SetParts(parts);  

    Aws::S3::Model::CompleteMultipartUploadRequest request;  

    request.SetBucket(bucket);  

    request.SetKey(key);  

    request.SetUploadId(uploadID);  

    request.SetMultipartUpload(completedMultipartUpload);  

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =  

        client.CompleteMultipartUpload(request);  

    if (!outcome.IsSuccess()) {  

        std::cerr << "Error completing multipart upload: " <<  

outcome.GetError().GetMessage() << std::endl;  

    }  

    return outcome;  

}

```

```

//! Routine which performs a multi-part upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashMethod: The hashing algorithm to use when calculating the hash
value.
    \param ioStream: An IOStream for the data to be uploaded.
    \param useDefaultHashMethod: A flag indicating whether to use the default
hash method or the one specified in the hashMethod parameter.
    \param[out] hashDataResult: The Hasher object that will store the
concatenated hash value.
    \param[out] partHashes: The vector that will store the calculated hash values
for each part of the file.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::doMultipartUpload(const Aws::String &bucket,
                                   const Aws::String &key,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   const std::shared_ptr<Aws::IOStream>
&ioStream,
                                   bool useDefaultHashMethod,
                                   AwsDoc::S3::Hasher &hashDataResult,
                                   std::vector<Aws::String> &partHashes,
                                   const Aws::S3::S3Client &client) {

    // Get object size.
    ioStream->seekg(0, ioStream->end);
    size_t objectSize = ioStream->tellg();
    ioStream->seekg(0, ioStream->beg);

    Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm =
    Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {
            checksumAlgorithm = getChecksumAlgorithmForHashMethod(hashMethod);
        }
    }
    Aws::String uploadID = createMultipartUpload(bucket, key, checksumAlgorithm,
    client);
    if (uploadID.empty()) {
        return false;
    }
}

```

```

std::vector<unsigned char> totalHashBuffer;
bool uploadSucceeded = true;
std::streamsize uploadedBytes = 0;
int partNumber = 1;
Aws::Vector<Aws::S3::Model::CompletedPart> parts;
while (uploadedBytes < objectSize) {
    std::cout << "Uploading part " << partNumber << "." << std::endl;

    std::vector<unsigned char> buffer(UPLOAD_BUFFER_SIZE);
    std::streamsize bytesToRead =
static_cast<std::streamsize>(std::min(buffer.size(),
objectSize - uploadedBytes));
    ioStream->read((char *) buffer.data(), bytesToRead);
    Aws::Utils::Stream::PreallocatedStreamBuf
preallocatedStreamBuf(buffer.data(),
bytesToRead);
    std::shared_ptr<Aws::IOStream> body =
        Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
            &preallocatedStreamBuf);

    Hasher hasher;
    if (!hasher.calculateObjectHash(*body, hashMethod)) {
        std::cerr << "Error calculating hash." << std::endl;
        uploadSucceeded = false;
        break;
    }

    Aws::String base64HashString = hasher.getBase64HashString();
    partHashes.push_back(base64HashString);

    Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

    totalHashBuffer.insert(totalHashBuffer.end(),
hashBuffer.GetUnderlyingData(),
                        hashBuffer.GetUnderlyingData() +
hashBuffer.GetLength());

    Aws::String calculatedHash;
    if (gUseCalculatedChecksum) {
        calculatedHash = base64HashString;
    }

```

```

        Aws::S3::Model::UploadPartOutcome uploadPartOutcome = uploadPart(bucket,
key, uploadID, partNumber,

checksumAlgorithm, base64HashString, body,

                                                                    client);

        if (uploadPartOutcome.IsSuccess()) {
            const Aws::S3::Model::UploadPartResult &uploadPartResult =
uploadPartOutcome.GetResult();
            Aws::S3::Model::CompletedPart completedPart;
            completedPart.SetETag(uploadPartResult.GetETag());
            completedPart.SetPartNumber(partNumber);
            switch (hashMethod) {
                case AwsDoc::S3::MD5:
                    break; // Do nothing.
                case AwsDoc::S3::SHA1:

completedPart.SetChecksumSHA1(uploadPartResult.GetChecksumSHA1());
                    break;
                case AwsDoc::S3::SHA256:

completedPart.SetChecksumSHA256(uploadPartResult.GetChecksumSHA256());
                    break;
                case AwsDoc::S3::CRC32:

completedPart.SetChecksumCRC32(uploadPartResult.GetChecksumCRC32());
                    break;
                case AwsDoc::S3::CRC32C:

completedPart.SetChecksumCRC32C(uploadPartResult.GetChecksumCRC32C());
                    break;
                default:
                    std::cerr << "Unhandled hash method for completedPart." <<
std::endl;
                    break;
            }

            parts.push_back(completedPart);
        } else {
            std::cerr << "Error uploading part. " <<
                uploadPartOutcome.GetError().GetMessage() << std::endl;
            uploadSucceeded = false;
            break;
        }
    }

```

```

        uploadedBytes += bytesToRead;
        partNumber++;
    }

    if (!uploadSucceeded) {
        abortMultipartUpload(bucket, key, uploadID, client);
        return false;
    } else {

        Aws::S3::Model::CompleteMultipartUploadOutcome
        completeMultipartUploadOutcome = completeMultipartUpload(bucket,

                                key,

                                uploadID,

                                parts,

                                client);

        if (completeMultipartUploadOutcome.IsSuccess()) {
            std::cout << "Multipart upload completed." << std::endl;
            if (!hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod))
        {
                std::cerr << "Error calculating hash." << std::endl;
                return false;
            }
        } else {
            std::cerr << "Error completing multipart upload." <<
                completeMultipartUploadOutcome.GetError().GetMessage()
                << std::endl;
        }

        return completeMultipartUploadOutcome.IsSuccess();
    }
}

//! Routine which retrieves the string for a HASH_METHOD constant.
/*!
    \param: hashMethod: A HASH_METHOD constant.
    \return: String: A string description of the hash method.
*/
Aws::String AwsDoc::S3::stringForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod) {
    switch (hashMethod) {

```

```

        case AwsDoc::S3::DEFAULT:
            return "Default";
        case AwsDoc::S3::MD5:
            return "MD5";
        case AwsDoc::S3::SHA1:
            return "SHA1";
        case AwsDoc::S3::SHA256:
            return "SHA256";
        case AwsDoc::S3::CRC32:
            return "CRC32";
        case AwsDoc::S3::CRC32C:
            return "CRC32C";
        default:
            return "Unknown";
    }
}

//! Routine that returns the ChecksumAlgorithm for a HASH_METHOD constant.
/*!
    \param: hashMethod: A HASH_METHOD constant.
    \return: ChecksumAlgorithm: The ChecksumAlgorithm enum.
*/
Aws::S3::Model::ChecksumAlgorithm
AwsDoc::S3::getChecksumAlgorithmForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod)
{
    Aws::S3::Model::ChecksumAlgorithm result =
    Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
    switch (hashMethod) {
        case AwsDoc::S3::DEFAULT:
            std::cerr << "getChecksumAlgorithmForHashMethod- DEFAULT is not
valid." << std::endl;
            break; // Default is not supported.
        case AwsDoc::S3::MD5:
            break; // Ignore MD5.
        case AwsDoc::S3::SHA1:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA1;
            break;
        case AwsDoc::S3::SHA256:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA256;
            break;
        case AwsDoc::S3::CRC32:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32;
            break;
        case AwsDoc::S3::CRC32C:

```

```

        result = Aws::S3::Model::ChecksumAlgorithm::CRC32C;
        break;
    default:
        std::cerr << "Unknown hash method." << std::endl;
        break;
    }

    return result;
}

//! Routine which cleans up after the example is complete.
/*!
    \param bucket: The name of the S3 bucket where the object was uploaded.
    \param clientConfiguration: The client configuration for the S3 client.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::cleanUp(const Aws::String &bucketName,
                        const Aws::S3::S3ClientConfiguration
                        &clientConfiguration) {

    Aws::Vector<Aws::String> keysResult;
    bool result = true;
    if (AwsDoc::S3::listObjects(bucketName, keysResult, clientConfiguration)) {
        if (!keysResult.empty()) {
            result = AwsDoc::S3::deleteObjects(keysResult, bucketName,
                                                clientConfiguration);
        }
    } else {
        result = false;
    }

    return result && AwsDoc::S3::deleteBucket(bucketName, clientConfiguration);
}

//! Console interaction introducing the workflow.
/*!
    \param bucketName: The name of the S3 bucket to use.
*/
void AwsDoc::S3::introductoryExplanations(const Aws::String &bucketName) {

    std::cout
        << "Welcome to the Amazon Simple Storage Service (Amazon S3) object
    integrity workflow."

```

```

        << std::endl;
    printAsterisksLine();
    std::cout
        << "This workflow demonstrates how Amazon S3 uses checksum values to
verify the integrity of data\n";
    std::cout << "uploaded to Amazon S3 buckets" << std::endl;
    std::cout
        << "The AWS SDK for C++ automatically handles checksums.\n";
    std::cout
        << "By default it calculates a checksum that is uploaded with an
object.\n"
        << "The default checksum algorithm for PutObject and MultiPart upload
is an MD5 hash.\n"
        << "The default checksum algorithm for TransferManager uploads is a
CRC32 checksum."
        << std::endl;
    std::cout
        << "You can override the default behavior, requiring one of the
following checksums,\n";
    std::cout << "MD5, CRC32, CRC32C, SHA-1 or SHA-256." << std::endl;
    std::cout << "You can also set the checksum hash value, instead of letting
the SDK calculate the value."
        << std::endl;
    std::cout
        << "For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/checking-object-integrity.html."
        << std::endl;

    std::cout
        << "This workflow will locally compute checksums for files uploaded
to an Amazon S3 bucket,\n";
    std::cout << "even when the SDK also computes the checksum." << std::endl;
    std::cout
        << "This is done to provide demonstration code for how the checksums
are calculated."
        << std::endl;
    std::cout << "A bucket named '" << bucketName << "' will be created for the
object uploads."
        << std::endl;
}

//! Console interaction which explains the PutObject results.
/*!
*/

```



```
void AwsDoc::S3::explainPutObjectResults() {

    std::cout << "The upload was successful.\n";
    std::cout << "If the checksums had not matched, the upload would have failed."
               << std::endl;
    std::cout
               << "The checksums calculated by the server have been retrieved using the GetObjectAttributes."
               << std::endl;
    std::cout
               << "The locally calculated checksums have been verified against the retrieved checksums."
               << std::endl;
}

//! Console interaction explaining transfer manager uploads.
/*!
\param objectKey: The key for the object being uploaded.
*/
void AwsDoc::S3::introductoryTransferManagerUploadExplanations(
    const Aws::String &objectKey) {
    std::cout
               << "Now the workflow will demonstrate object integrity for TransferManager multi-part uploads."
               << std::endl;
    std::cout
               << "The AWS C++ SDK has a TransferManager class which simplifies multipart uploads."
               << std::endl;
    std::cout
               << "The following code lets the TransferManager handle much of the checksum configuration."
               << std::endl;

    std::cout << "An object with the key '" << objectKey
               << " will be uploaded by the TransferManager using a "
               << BUFFER_SIZE_IN_MEGABYTES << " MB buffer." << std::endl;
    if (gUseCalculatedChecksum) {
        std::cout << "For TransferManager uploads, this demo always lets the SDK calculate the hash value."
                   << std::endl;
    }
}
```

```

        pressEnterToContinue();
        printAsterisksLine();
    }

    //! Console interaction explaining multi-part uploads.
    /*!
        \param objectKey: The key for the object being uploaded.
        \param chosenHashMethod: The hash method selected by the user.
    */
    void AwsDoc::S3::multiPartUploadExplanations(const Aws::String &objectKey,
                                                HASH_METHOD chosenHashMethod) {

        std::cout
            << "Now we will provide an in-depth demonstration of multi-part
            uploading by calling the multi-part upload APIs directly."
            << std::endl;

        std::cout << "These are the same APIs used by the TransferManager when
            uploading large files."
            << std::endl;

        std::cout
            << "In the following code, the checksums are also calculated locally
            and then compared."
            << std::endl;

        std::cout
            << "For multi-part uploads, a checksum is uploaded with each part.
            The final checksum is a concatenation of"
            << std::endl;

        std::cout << "the checksums for each part." << std::endl;

        std::cout
            << "This is explained in the user guide, https://docs.aws.amazon.com/
            AmazonS3/latest/userguide/checking-object-integrity.html,"
            << " in the section \"Using part-level checksums for multipart
            uploads\"." << std::endl;

        std::cout << "Starting multipart upload of with hash method " <<
            stringForHashMethod(chosenHashMethod) << " uploading to with object
            key\n"
            << "\"" << objectKey << "\", " << std::endl;

    }

    //! Create a large file for doing multi-part uploads.
    /*!
    */
    bool AwsDoc::S3::createLargeFileIfNotExists() {

```

```

// Generate a large file by writing this source file multiple times to a new
file.
if (std::filesystem::exists(MULTI_PART_TEST_FILE)) {
    return true;
}

std::ofstream newFile(MULTI_PART_TEST_FILE, std::ios::out

                                | std::ios::binary);

if (!newFile) {
    std::cerr << "createLargeFileIfNotExists- Error creating file " <<
MULTI_PART_TEST_FILE <<
        std::endl;
    return false;
}

std::ifstream input(TEST_FILE, std::ios::in

                                | std::ios::binary);

if (!input) {
    std::cerr << "Error opening file " << TEST_FILE <<
        std::endl;
    return false;
}
std::stringstream buffer;
buffer << input.rdbuf();

input.close();

while (newFile.tellp() < LARGE_FILE_SIZE && !newFile.bad()) {
    buffer.seekg(std::stringstream::beg);
    newFile << buffer.rdbuf();
}

newFile.close();

return true;
}

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per C++ .

- [AbortMultipartUpload](#)
- [CompleteMultipartUpload](#)
- [CreateMultipartUpload](#)
- [DeleteObject](#)
- [GetObjectAttributes](#)
- [PutObject](#)
- [UploadPart](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Lavora con oggetti con versione di Amazon S3 utilizzando un AWS SDK

L'esempio di codice seguente mostra come:

- Creazione un bucket S3 con versione.
- Ottenimento di tutte le versioni di un oggetto.
- Ripristino di un oggetto a una versione precedente.
- Eliminazione e ripristino di un oggetto con versione.
- Eliminazione permanente di tutte le versioni di un oggetto.

Python

SDK per Python (Boto3)

Note

C'è altro su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Creazione di funzioni che eseguono il wrap delle operazioni S3.

```
def create_versioned_bucket(bucket_name, prefix):  
    """
```

Creates an Amazon S3 bucket, enables it for versioning, and configures a lifecycle

that expires noncurrent object versions after 7 days.

Adding a lifecycle configuration to a versioned bucket is a best practice. It helps prevent objects in the bucket from accumulating a large number of noncurrent versions, which can slow down request performance.

Usage is shown in the `usage_demo_single_object` function at the end of this module.

```
:param bucket_name: The name of the bucket to create.
:param prefix: Identifies which objects are automatically expired under the
               configured lifecycle rules.
:return: The newly created bucket.
"""
try:
    bucket = s3.create_bucket(
        Bucket=bucket_name,
        CreateBucketConfiguration={
            "LocationConstraint": s3.meta.client.meta.region_name
        },
    )
    logger.info("Created bucket %s.", bucket.name)
except ClientError as error:
    if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
        logger.warning("Bucket %s already exists! Using it.", bucket_name)
        bucket = s3.Bucket(bucket_name)
    else:
        logger.exception("Couldn't create bucket %s.", bucket_name)
        raise

try:
    bucket.Versioning().enable()
    logger.info("Enabled versioning on bucket %s.", bucket.name)
except ClientError:
    logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
    raise

try:
    expiration = 7
    bucket.LifecycleConfiguration().put(
        LifecycleConfiguration={
            "Rules": [
```

```

        {
            "Status": "Enabled",
            "Prefix": prefix,
            "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
        }
    ]
}
)
logger.info(
    "Configured lifecycle to expire noncurrent versions after %s days "
    "on bucket %s.",
    expiration,
    bucket.name,
)
except ClientError as error:
    logger.warning(
        "Couldn't configure lifecycle on bucket %s because %s. "
        "Continuing anyway.",
        bucket.name,
        error,
    )

return bucket

```

```
def rollback_object(bucket, object_key, version_id):
```

```
    """
```

Rolls back an object to an earlier version by deleting all versions that occurred after the specified rollback version.

Usage is shown in the `usage_demo_single_object` function at the end of this module.

```
    :param bucket: The bucket that holds the object to roll back.
```

```
    :param object_key: The object to roll back.
```

```
    :param version_id: The version ID to roll back to.
```

```
    """
```

```
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
```

```
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
    )

```

```

        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
            f"{object_key}."
        )

def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
    marker.
    By removing the delete marker, we make the previous version the latest
    version
    and the object then presents as not deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

```

```

:param bucket: The bucket that contains the object.
:param object_key: The object to revive.
"""

# Get the latest version for the object.
response = s3.meta.client.list_object_versions(
    Bucket=bucket.name, Prefix=object_key, MaxKeys=1
)

if "DeleteMarkers" in response:
    latest_version = response["DeleteMarkers"][0]
    if latest_version["IsLatest"]:
        logger.info(
            "Object %s was indeed deleted on %s. Let's revive it.",
            object_key,
            latest_version["LastModified"],
        )
        obj = bucket.Object(object_key)
        obj.Version(latest_version["VersionId"]).delete()
        logger.info(
            "Revived %s, active version is now %s with body '%s'",
            object_key,
            obj.version_id,
            obj.get()["Body"].read(),
        )
    else:
        logger.warning(
            "Delete marker is not the latest version for %s!", object_key
        )
elif "Versions" in response:
    logger.warning("Got an active version for %s, nothing to do.",
object_key)
else:
    logger.error("Couldn't get any version info for %s.", object_key)

def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that contains the object.

```



```

:param object_key: The object to delete.
"""
try:
    bucket.object_versions.filter(Prefix=object_key).delete()
    logger.info("Permanently deleted all versions of object %s.", object_key)
except ClientError:
    logger.exception("Couldn't delete all versions of %s.", object_key)
    raise

```

Caricamento la strofa di una poesia in un oggetto con versione ed esecuzione di una serie di operazioni su di esso.

```

def usage_demo_single_object(obj_prefix="demo-versioning/"):
    """
    Demonstrates usage of versioned object functions. This demo uploads a stanza
    of a poem and performs a series of revisions, deletions, and revivals on it.

    :param obj_prefix: The prefix to assign to objects created by this demo.
    """
    with open("father_william.txt") as file:
        stanzas = file.read().split("\n\n")

    width = get_terminal_size((80, 20))[0]
    print("-" * width)
    print("Welcome to the usage demonstration of Amazon S3 versioning.")
    print(
        "This demonstration uploads a single stanza of a poem to an Amazon "
        "S3 bucket and then applies various revisions to it."
    )
    print("-" * width)
    print("Creating a version-enabled bucket for the demo...")
    bucket = create_versioned_bucket("bucket-" + str(uuid.uuid1()), obj_prefix)

    print("\nThe initial version of our stanza:")
    print(stanzas[0])

    # Add the first stanza and revise it a few times.
    print("\nApplying some revisions to the stanza...")
    obj_stanza_1 = bucket.Object(f"{obj_prefix}stanza-1")
    obj_stanza_1.put(Body=bytes(stanzas[0], "utf-8"))

```

```

obj_stanza_1.put(Body=bytes(stanzas[0].upper(), "utf-8"))
obj_stanza_1.put(Body=bytes(stanzas[0].lower(), "utf-8"))
obj_stanza_1.put(Body=bytes(stanzas[0][::-1], "utf-8"))
print(
    "The latest version of the stanza is now:",
    obj_stanza_1.get()["Body"].read().decode("utf-8"),
    sep="\n",
)

# Versions are returned in order, most recent first.
obj_stanza_1_versions =
bucket.object_versions.filter(Prefix=obj_stanza_1.key)
print(
    "The version data of the stanza revisions:",
    *[
        f"    {version.version_id}, last modified {version.last_modified}"
        for version in obj_stanza_1_versions
    ],
    sep="\n",
)

# Rollback two versions.
print("\nRolling back two versions...")
rollback_object(bucket, obj_stanza_1.key, list(obj_stanza_1_versions)
[2].version_id)
print(
    "The latest version of the stanza:",
    obj_stanza_1.get()["Body"].read().decode("utf-8"),
    sep="\n",
)

# Delete the stanza
print("\nDeleting the stanza...")
obj_stanza_1.delete()
try:
    obj_stanza_1.get()
except ClientError as error:
    if error.response["Error"]["Code"] == "NoSuchKey":
        print("The stanza is now deleted (as expected).")
    else:
        raise

# Revive the stanza
print("\nRestoring the stanza...")

```

```
    revive_object(bucket, obj_stanza_1.key)
    print(
        "The stanza is restored! The latest version is again:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Permanently delete all versions of the object. This cannot be undone!
    print("\nPermanently deleting all versions of the stanza...")
    permanently_delete_object(bucket, obj_stanza_1.key)
    obj_stanza_1_versions =
bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    if len(list(obj_stanza_1_versions)) == 0:
        print("The stanza has been permanently deleted and now has no versions.")
    else:
        print("Something went wrong. The stanza still exists!")

    print(f"\nRemoving {bucket.name}...")
    bucket.delete()
    print(f"{bucket.name} deleted.")
    print("Demo done!")
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Python (Boto3).
 - [CreateBucket](#)
 - [DeleteObject](#)
 - [ListObjectVersions](#)
 - [PutBucketLifecycleConfiguration](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Esempi serverless per Amazon S3

I seguenti esempi di codice mostrano come usare Amazon S3 con AWS SDK.

Esempi

- [Invocazione di una funzione Lambda da un trigger Amazon S3](#)

Invocazione di una funzione Lambda da un trigger Amazon S3

I seguenti esempi di codice mostrano come implementare una funzione Lambda che riceve un evento attivato dal caricamento di un oggetto in un bucket S3. La funzione recupera il nome del bucket S3 e la chiave dell'oggetto dal parametro evento e chiama l'API Amazon S3 per recuperare e registrare il tipo di contenuto dell'oggetto.

.NET

SDK per .NET

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Utilizzo di un evento S3 con Lambda tramite .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Threading.Tasks;
using Amazon.Lambda.Core;
using Amazon.S3;
using System;
using Amazon.Lambda.S3Events;
using System.Web;

// Assembly attribute to enable the Lambda function's JSON input to be converted
// into a .NET class.
[assembly:
    LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSerializer))]

namespace S3Integration
{
    public class Function
    {
        private static AmazonS3Client _s3Client;
```

```
public Function() : this(null)
{
}

internal Function(AmazonS3Client s3Client)
{
    _s3Client = s3Client ?? new AmazonS3Client();
}

public async Task<string> Handler(S3Event evt, ILambdaContext context)
{
    try
    {
        if (evt.Records.Count <= 0)
        {
            context.Logger.LogLine("Empty S3 Event received");
            return string.Empty;
        }

        var bucket = evt.Records[0].S3.Bucket.Name;
        var key = HttpUtility.UrlDecode(evt.Records[0].S3.Object.Key);

        context.Logger.LogLine($"Request is for {bucket} and {key}");

        var objectResult = await _s3Client.GetObjectAsync(bucket, key);

        context.Logger.LogLine($"Returning {objectResult.Key}");

        return objectResult.Key;
    }
    catch (Exception e)
    {
        context.Logger.LogLine($"Error processing request -
{e.Message}");

        return string.Empty;
    }
}
}
```

Go

SDK per Go V2

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Utilizzo di un evento S3 con Lambda tramite Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
    "log"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)

func handler(ctx context.Context, s3Event events.S3Event) error {
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        log.Printf("failed to load default config: %s", err)
        return err
    }
    s3Client := s3.NewFromConfig(sdkConfig)

    for _, record := range s3Event.Records {
        bucket := record.S3.Bucket.Name
        key := record.S3.Object.URLDecodedKey
        headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{
            Bucket: &bucket,
            Key:    &key,
        })
        if err != nil {
            log.Printf("error getting head of object %s/%s: %s", bucket, key, err)
        }
    }
}
```

```
    return err
}
log.Printf("successfully retrieved %s/%s of type %s", bucket, key,
*headOutput.ContentType)
}

return nil
}

func main() {
    lambda.Start(handler)
}
```

Java

SDK per Java 2.x

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Utilizzo di un evento S3 con Lambda tramite Java.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
    com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotifi

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
            S3EventNotificationRecord record = s3event.getRecords().get(0);
            String srcBucket = record.getS3().getBucket().getName();
            String srcKey = record.getS3().getObject().getUrlDecodedKey();

            S3Client s3Client = S3Client.builder().build();
            HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
srcKey);

            logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
type " + headObject.contentType());

            return "Ok";
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucket)
            .key(key)
            .build();
        return s3Client.headObject(headObjectRequest);
    }
}

```

JavaScript

SDK per JavaScript (v3)

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Consumo di un evento S3 con JavaScript Lambda utilizzando.

```
import { S3Client, HeadObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client();

export const handler = async (event, context) => {

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, ' '));

    try {
        const { ContentType } = await client.send(new HeadObjectCommand({
            Bucket: bucket,
            Key: key,
        }));

        console.log('CONTENT TYPE:', ContentType);
        return ContentType;

    } catch (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make sure they exist and your bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
    }
};
```

Consumo di un evento S3 con TypeScript Lambda utilizzando.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Event } from 'aws-lambda';
import { S3Client, HeadObjectCommand } from '@aws-sdk/client-s3';

const s3 = new S3Client({ region: process.env.AWS_REGION });

export const handler = async (event: S3Event): Promise<string | undefined> => {
    // Get the object from the event and show its content type
```

```
const bucket = event.Records[0].s3.bucket.name;
const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, ' '));
const params = {
  Bucket: bucket,
  Key: key,
};
try {
  const { ContentType } = await s3.send(new HeadObjectCommand(params));
  console.log('CONTENT TYPE:', ContentType);
  return ContentType;
} catch (err) {
  console.log(err);
  const message = `Error getting object ${key} from bucket ${bucket}. Make sure they exist and your bucket is in the same region as this function.`;
  console.log(message);
  throw new Error(message);
}
};
```

PHP

SDK per PHP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Utilizzo di un evento S3 con Lambda tramite PHP.

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';
```

```
class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' .
                $bucket . '. Make sure they exist and your bucket is in the same region as this
                function.' . "\n";
                throw $e;
            }
        }
    }
}

$logger = new StderrLogger();
return new Handler($logger);
```

Python

SDK per Python (Boto3)

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Utilizzo di un evento S3 con Lambda tramite Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
import json
import urllib.parse
import boto3

print('Loading function')

s3 = boto3.client('s3')

def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],
    encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("CONTENT TYPE: " + response['ContentType'])
        return response['ContentType']
    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and
        your bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

Ruby

SDK per Ruby

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Utilizzo di un evento S3 con Lambda tramite Ruby.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
  Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
    and your bucket is in the same region as this function."
    raise e
  end
end
```

Rust

SDK per Rust

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri come eseguire la configurazione e l'esecuzione nel repository di [Esempi serverless](#).

Utilizzo di un evento S3 con Lambda tramite Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::s3::S3Event;
use aws_sdk_s3::{Client};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

/// Main function
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    // Initialize the AWS SDK for Rust
    let config = aws_config::load_from_env().await;
    let s3_client = Client::new(&config);

    let res = run(service_fn(|request: LambdaEvent<S3Event>| {
        function_handler(&s3_client, request)
    })).await;

    res
}

async fn function_handler(
    s3_client: &Client,
    evt: LambdaEvent<S3Event>
) -> Result<(), Error> {
```

```
tracing::info!(records = ?evt.payload.records.len(), "Received request from
SQS");

if evt.payload.records.len() == 0 {
    tracing::info!("Empty S3 event received");
}

let bucket = evt.payload.records[0].s3.bucket.name.as_ref().expect("Bucket
name to exist");
let key = evt.payload.records[0].s3.object.key.as_ref().expect("Object key to
exist");

tracing::info!("Request is for {} and object {}", bucket, key);

let s3_get_object_result = s3_client
    .get_object()
    .bucket(bucket)
    .key(key)
    .send()
    .await;

match s3_get_object_result {
    Ok(_) => tracing::info!("S3 Get Object success, the s3GetObjectResult
contains a 'body' property of type ByteStream"),
    Err(_) => tracing::info!("Failure with S3 Get Object request")
}

Ok(())
}
```

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Esempi di codice per l'utilizzo di Amazon S3 Control AWS SDK

I seguenti esempi di codice mostrano come usare Amazon S3 Control con un kit di sviluppo AWS software (SDK).

Nozioni di base: esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Per un elenco completo di guide ed esempi di codice per sviluppatori AWS SDK, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Amazon S3 Control

- [Esempi di base per l'utilizzo di Amazon S3 Control AWS SDK](#)
 - [Hello Amazon S3 Control](#)
 - [Scopri le nozioni di base di Amazon S3 Control con un AWS SDK](#)
 - [Azioni per Amazon S3 Control utilizzando AWS SDK](#)
 - [Da utilizzare CreateJobcon un AWS SDK o CLI](#)
 - [Utilizzare DeleteJobTaggingcon un AWS SDK](#)
 - [Utilizzare DescribeJobcon un AWS SDK o CLI](#)
 - [Utilizzare GetJobTaggingcon un AWS SDK](#)
 - [Utilizzare PutJobTaggingcon un AWS SDK](#)
 - [Utilizzare UpdateJobPrioritycon un AWS SDK o CLI](#)
 - [Utilizzare UpdateJobStatuscon un AWS SDK o CLI](#)

Esempi di base per l'utilizzo di Amazon S3 Control AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare le nozioni di base di Amazon S3 Control con gli SDK AWS .

Esempi

- [Hello Amazon S3 Control](#)
- [Scopri le nozioni di base di Amazon S3 Control con un AWS SDK](#)
- [Azioni per Amazon S3 Control utilizzando AWS SDK](#)
 - [Da utilizzare CreateJobcon un AWS SDK o CLI](#)
 - [Utilizzare DeleteJobTaggingcon un AWS SDK](#)
 - [Utilizzare DescribeJobcon un AWS SDK o CLI](#)
 - [Utilizzare GetJobTaggingcon un AWS SDK](#)

- [Utilizzare PutJobTagging con un AWS SDK](#)
- [Utilizzare UpdateJobPriority con un AWS SDK o CLI](#)
- [Utilizzare UpdateJobStatus con un AWS SDK o CLI](#)

Hello Amazon S3 Control

I seguenti esempi di codice mostrano come iniziare a usare Amazon S3 Control.

Java

SDK per Java 2.x

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlAsyncClient;
import software.amazon.awssdk.services.s3control.model.JobListDescriptor;
import software.amazon.awssdk.services.s3control.model.JobStatus;
import software.amazon.awssdk.services.s3control.model.ListJobsRequest;
import software.amazon.awssdk.services.s3control.paginators.ListJobsPublisher;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this example:
 * <p/>
```

```

* The SDK must be able to authenticate AWS requests on your behalf. If you have
not configured
* authentication for SDKs and tools, see https://docs.aws.amazon.com/sdkref/latest/guide/access.html in the AWS SDKs and Tools Reference Guide.
* <p/>
* You must have a runtime environment configured with the Java SDK.
* See https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html
in the Developer Guide if this is not set up.
*/
public class HelloS3Batch {
    private static S3ControlAsyncClient asyncClient;

    public static void main(String[] args) {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        try {
            listBatchJobsAsync(accountId)
                .exceptionally(ex -> {
                    System.err.println("List batch jobs failed: " +
ex.getMessage());
                    return null;
                })
                .join();

        } catch (CompletionException ex) {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
        }
    }

    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
S3ControlAsyncClient}. If the instance
     * has not been created yet, it will be initialized with the following
configuration:
     * <ul>
     * <li>Maximum concurrency: 100</li>
     * <li>Connection timeout: 60 seconds</li>
     * <li>Read timeout: 60 seconds</li>
     * <li>Write timeout: 60 seconds</li>
     * <li>API call timeout: 2 minutes</li>
     * <li>API call attempt timeout: 90 seconds</li>
     * <li>Retry policy: 3 retries</li>

```

```

    * <li>Region: US_EAST_1</li>
    * <li>Credentials provider: {@link
EnvironmentVariableCredentialsProvider}</li>
    * </ul>
    *
    * @return the asynchronous S3 Control client instance
    */
    private static S3ControlAsyncClient getAsyncClient() {
        if (asyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))
                .readTimeout(Duration.ofSeconds(60))
                .writeTimeout(Duration.ofSeconds(60))
                .build();

            ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
                .apiCallTimeout(Duration.ofMinutes(2))
                .apiCallAttemptTimeout(Duration.ofSeconds(90))
                .retryStrategy(RetryMode.STANDARD)
                .build();

            asyncClient = S3ControlAsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
        return asyncClient;
    }

    /**
     * Asynchronously lists batch jobs that have completed for the specified
     * account.
     *
     * @param accountId the ID of the account to list jobs for
     * @return a CompletableFuture that completes when the job listing operation
     * is finished
     */
    public static CompletableFuture<Void> listBatchJobsAsync(String accountId) {
        ListJobsRequest jobsRequest = ListJobsRequest.builder()
            .jobStatuses(JobStatus.COMPLETE)
            .accountId(accountId)

```

```

        .maxResults(10)
        .build();

    ListJobsPublisher publisher =
getAsyncClient().listJobsPaginator(jobsRequest);
    return publisher.subscribe(response -> {
        List<JobListDescriptor> jobs = response.jobs();
        for (JobListDescriptor job : jobs) {
            System.out.println("The job id is " + job.jobId());
            System.out.println("The job priority is " + job.priority());
        }
    }).thenAccept(response -> {
        System.out.println("Listing batch jobs completed");
    }).exceptionally(ex -> {
        System.err.println("Failed to list batch jobs: " + ex.getMessage());
        throw new RuntimeException(ex);
    });
}

```

- Per i dettagli sull'API, consulta la [ListJobs](#) sezione AWS SDK for Java 2.x API Reference.

Python

SDK per Python (Boto3)

Note

C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

def list_jobs(self, account_id: str) -> None:
    """
    List all batch jobs for the account.

    Args:
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.list_jobs(
            AccountId=account_id,

```

```
        JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']
    )
    jobs = response.get('Jobs', [])
    for job in jobs:
        print(f"The job id is {job['JobId']}")
        print(f"The job priority is {job['Priority']}")
except ClientError as e:
    print(f"Error listing jobs: {e}")
    raise
```

- Per i dettagli sull'API, consulta [ListJobs AWSSDK](#) for Python (Boto3) API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Scopri le nozioni di base di Amazon S3 Control con un AWS SDK

I seguenti esempi di codice mostrano come apprendere le operazioni di base per Amazon S3 Control.

Java

SDK per Java 2.x

Note

C'è altro su [GitHub](#). Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Nozioni di base sulle operazioni principali.

```
package com.example.s3.batch;

import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.IOException;
import java.util.Map;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletionException;
```

```
public class S3BatchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
    "-");
    private static final String STACK_NAME = "MyS3Stack";
    public static void main(String[] args) throws IOException {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        String uuid = java.util.UUID.randomUUID().toString();
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon S3 Batch basics scenario.");
        System.out.println("""
            S3 Batch operations enables efficient and cost-effective processing
of large-scale
            data stored in Amazon S3. It automatically scales resources to handle
varying workloads
            without the need for manual intervention.

            One of the key features of S3 Batch is its ability to perform tagging
operations on objects stored in
            S3 buckets. Users can leverage S3 Batch to apply, update, or remove
tags on thousands or millions of
            objects in a single operation, streamlining the management and
organization of their data.

            This can be particularly useful for tasks such as cost allocation,
lifecycle management, or
            metadata-driven workflows, where consistent and accurate tagging is
essential.

            S3 Batch's scalability and serverless nature make it an ideal
solution for organizations with
            growing data volumes and complex data management requirements.

            This Java program walks you through Amazon S3 Batch operations.

            Let's get started...

            """);
        waitForInputToContinue(scanner);
        // Use CloudFormation to stand up the resource required for this
scenario.
```

```
        System.out.println("Use CloudFormation to stand up the resource required
for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME);

        Map<String, String> stackOutputs =
CloudFormationHelper.getStackOutputs(STACK_NAME);
        String iamRoleArn = stackOutputs.get("S3BatchRoleArn");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Setup the required bucket for this scenario.");
        waitForInputToContinue(scanner);
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
name.
        actions.createBucket(bucketName);
        String reportBucketName = "arn:aws:s3:::"+bucketName;
        String manifestLocation = "arn:aws:s3:::"+bucketName+"/job-manifest.csv";
        System.out.println("Populate the bucket with the required files.");
        String[] fileNames = {"job-manifest.csv", "object-key-1.txt", "object-
key-2.txt", "object-key-3.txt", "object-key-4.txt"};
        actions.uploadFilesToBucket(bucketName, fileNames, actions);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a S3 Batch Job");
        System.out.println("This job tags all objects listed in the manifest file
with tags");
        waitForInputToContinue(scanner);
        String jobId ;
        try {
            jobId = actions.createS3JobAsync(accountId, iamRoleArn,
manifestLocation, reportBucketName, uuid).join();
            System.out.println("The Job id is " + jobId);

        } catch (S3Exception e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
```

```

        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Update an existing S3 Batch Operations job's
priority");
        System.out.println(""""
            In this step, we modify the job priority value. The higher the
number, the higher the priority.
            So, a job with a priority of `30` would have a higher priority than
a job with
            a priority of `20`. This is a common way to represent the priority
of a task
            or job, with higher numbers indicating a higher priority.

            Ensure that the job status allows for priority updates. Jobs in
certain
            states (e.g., Cancelled, Failed, or Completed) cannot have their
priorities
            updated. Only jobs in the Active or Suspended state typically allow
priority
            updates.
            """);

        try {
            actions.updateJobPriorityAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Update job priority failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to update job priority: " +
ex.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Cancel the S3 Batch job");
        System.out.print("Do you want to cancel the Batch job? (y/n): ");
        String cancelAns = scanner.nextLine();
        if (cancelAns != null && cancelAns.trim().equalsIgnoreCase("y")) {
            try {

```



```
        actions.cancelJobAsync(jobId, accountId)
            .exceptionally(ex -> {
                System.err.println("Cancel job failed: " +
ex.getMessage());
                return null;
            })
            .join();
    } catch (CompletionException ex) {
        System.err.println("Failed to cancel job: " + ex.getMessage());
    }
} else {
    System.out.println("Job " + jobId + " was not canceled.");
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Describe the job that was just created");
waitForInputToContinue(scanner);
try {
    actions.describeJobAsync(jobId, accountId)
        .exceptionally(ex -> {
            System.err.println("Describe job failed: " +
ex.getMessage());
            return null;
        })
        .join();
} catch (CompletionException ex) {
    System.err.println("Failed to describe job: " + ex.getMessage());
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Describe the tags associated with the job");
waitForInputToContinue(scanner);
try {
    actions.getJobTagsAsync(jobId, accountId)
        .exceptionally(ex -> {
            System.err.println("Get job tags failed: " +
ex.getMessage());
            return null;
        })
        .join();
} catch (CompletionException ex) {
    System.err.println("Failed to get job tags: " + ex.getMessage());
}
```

```
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("6. Update Batch Job Tags");
    waitForInputToContinue(scanner);
    try {
        actions.putJobTaggingAsync(jobId, accountId)
            .exceptionally(ex -> {
                System.err.println("Put job tagging failed: " +
ex.getMessage());
                return null;
            })
            .join();
    } catch (CompletionException ex) {
        System.err.println("Failed to put job tagging: " + ex.getMessage());
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("7. Delete the Amazon S3 Batch job tagging.");
    System.out.print("Do you want to delete Batch job tagging? (y/n)");
    String delAns = scanner.nextLine();
    if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
        try {
            actions.deleteBatchJobTagsAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Delete batch job tags failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to delete batch job tags: " +
ex.getMessage());
        }
    } else {
        System.out.println("Tagging was not deleted.");
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.print("Do you want to delete the AWS resources used in this
scenario? (y/n)");
```

```
String delResAns = scanner.nextLine();
if (delResAns != null && delResAns.trim().equalsIgnoreCase("y")) {
    actions.deleteFilesFromBucket(bucketName, fileNames, actions);
    actions.deleteBucketFolderAsync(bucketName);
    actions.deleteBucket(bucketName)
        .thenRun(() -> System.out.println("Bucket deletion completed"))
        .exceptionally(ex -> {
            System.err.println("Error occurred: " + ex.getMessage());
            return null;
        });
    CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
} else {
    System.out.println("The AWS resources were not deleted.");
}
System.out.println("The Amazon S3 Batch scenario has successfully
completed.");
System.out.println(DASHES);
}

private static void waitForInputToContinue(Scanner scanner) {
    while (true) {
        System.out.println();
        System.out.println("Enter 'c' followed by <ENTER> to continue:");
        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
            System.out.println("Continuing with the program...");
            System.out.println();
            break;
        } else {
            // Handle invalid input.
            System.out.println("Invalid input. Please try again.");
        }
    }
}
}
```

Una classe di azione per il wrapping delle operazioni.

```
public class S3BatchActions {
```

```

private static S3ControlAsyncClient asyncClient;

private static S3AsyncClient s3AsyncClient ;
/**
 * Retrieves the asynchronous S3 Control client instance.
 * <p>
 * This method creates and returns a singleton instance of the {@link
S3ControlAsyncClient}. If the instance
 * has not been created yet, it will be initialized with the following
configuration:
 * <ul>
 * <li>Maximum concurrency: 100</li>
 * <li>Connection timeout: 60 seconds</li>
 * <li>Read timeout: 60 seconds</li>
 * <li>Write timeout: 60 seconds</li>
 * <li>API call timeout: 2 minutes</li>
 * <li>API call attempt timeout: 90 seconds</li>
 * <li>Retry policy: 3 retries</li>
 * <li>Region: US_EAST_1</li>
 * <li>Credentials provider: {@link
EnvironmentVariableCredentialsProvider}</li>
 * </ul>
 *
 * @return the asynchronous S3 Control client instance
 */
private static S3ControlAsyncClient getAsyncClient() {
    if (asyncClient == null) {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();
    }
}

```

```

        asyncClient = S3ControlAsyncClient.builder()
            .region(Region.US_EAST_1)
            .httpClient(httpClient)
            .overrideConfiguration(overrideConfig)
            .build();
    }
    return asyncClient;
}

private static S3AsyncClient getS3AsyncClient() {
    if (asyncClient == null) {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryStrategy(RetryMode.STANDARD)
            .build();

        s3AsyncClient = S3AsyncClient.builder()
            .region(Region.US_EAST_1)
            .httpClient(httpClient)
            .overrideConfiguration(overrideConfig)
            .build();
    }
    return s3AsyncClient;
}

/**
 * Cancels a job asynchronously.
 *
 * @param jobId The ID of the job to be canceled.
 * @param accountId The ID of the account associated with the job.
 * @return A {@link CompletableFuture} that completes when the job status has
been updated to "CANCELLED".
 *
 * If an error occurs during the update, the returned future will
complete exceptionally.

```

```
    */
    public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId)
    {
        UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
            .build();

        return asyncClient.updateJobStatus(updateJobStatusRequest)
            .thenAccept(updateJobStatusResponse -> {
                System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
            })
            .exceptionally(ex -> {
                System.err.println("Failed to cancel job: " + ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }

    /**
     * Updates the priority of a job asynchronously.
     *
     * @param jobId      the ID of the job to update
     * @param accountId the ID of the account associated with the job
     * @return a {@link CompletableFuture} that represents the asynchronous
operation, which completes when the job priority has been updated or an error
has occurred
     */
    public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
        UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .priority(60)
            .build();

        CompletableFuture<Void> future = new CompletableFuture<>();
        getAsyncClient().updateJobPriority(priorityRequest)
            .thenAccept(response -> {
                System.out.println("The job priority was updated");
            });
    }
}
```

```
        future.complete(null); // Complete the CompletableFuture on
successful execution
    })
    .exceptionally(ex -> {
        System.err.println("Failed to update job priority: " +
ex.getMessage());
        future.completeExceptionally(ex); // Complete the
CompletableFuture exceptionally on error
        return null; // Return null to handle the exception
    });

    return future;
}

/**
 * Asynchronously retrieves the tags associated with a specific job in an AWS
account.
 *
 * @param jobId      the ID of the job for which to retrieve the tags
 * @param accountId the ID of the AWS account associated with the job
 * @return a {@link CompletableFuture} that completes when the job tags have
been retrieved, or with an exception if the operation fails
 * @throws RuntimeException if an error occurs while retrieving the job tags
 */
public CompletableFuture<Void> getJobTagsAsync(String jobId, String
accountId) {
    GetJobTaggingRequest request = GetJobTaggingRequest.builder()
        .jobId(jobId)
        .accountId(accountId)
        .build();

    return asyncClient.getJobTagging(request)
        .thenAccept(response -> {
            List<S3Tag> tags = response.tags();
            if (tags.isEmpty()) {
                System.out.println("No tags found for job ID: " + jobId);
            } else {
                for (S3Tag tag : tags) {
                    System.out.println("Tag key is: " + tag.key());
                    System.out.println("Tag value is: " + tag.value());
                }
            }
        })
        .exceptionally(ex -> {
```

```

        System.err.println("Failed to get job tags: " + ex.getMessage());
        throw new RuntimeException(ex); // Propagate the exception
    });
}

/**
 * Asynchronously deletes the tags associated with a specific batch job.
 *
 * @param jobId      The ID of the batch job whose tags should be deleted.
 * @param accountId  The ID of the account associated with the batch job.
 * @return A CompletableFuture that completes when the job tags have been
 *         successfully deleted, or an exception is thrown if the deletion fails.
 */
public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
    DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
        .accountId(accountId)
        .jobId(jobId)
        .build();

    return asyncClient.deleteJobTagging(jobTaggingRequest)
        .thenAccept(response -> {
            System.out.println("You have successfully deleted " + jobId + "
tagging.");
        })
        .exceptionally(ex -> {
            System.err.println("Failed to delete job tags: " +
ex.getMessage());
            throw new RuntimeException(ex);
        });
}

/**
 * Asynchronously describes the specified job.
 *
 * @param jobId      the ID of the job to describe
 * @param accountId  the ID of the AWS account associated with the job
 * @return a {@link CompletableFuture} that completes when the job
 *         description is available
 * @throws RuntimeException if an error occurs while describing the job
 */
public CompletableFuture<Void> describeJobAsync(String jobId, String
accountId) {

```



```
DescribeJobRequest jobRequest = DescribeJobRequest.builder()
    .jobId(jobId)
    .accountId(accountId)
    .build();

return getAsyncClient().describeJob(jobRequest)
    .thenAccept(response -> {
        System.out.println("Job ID: " + response.job().jobId());
        System.out.println("Description: " +
response.job().description());
        System.out.println("Status: " + response.job().statusAsString());
        System.out.println("Role ARN: " + response.job().roleArn());
        System.out.println("Priority: " + response.job().priority());
        System.out.println("Progress Summary: " +
response.job().progressSummary());

        // Print out details about the job manifest.
        JobManifest manifest = response.job().manifest();
        System.out.println("Manifest Location: " +
manifest.location().objectArn());
        System.out.println("Manifest ETag: " +
manifest.location().eTag());

        // Print out details about the job operation.
        JobOperation operation = response.job().operation();
        if (operation.s3PutObjectTagging() != null) {
            System.out.println("Operation: S3 Put Object Tagging");
            System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
        }

        // Print out details about the job report.
        JobReport report = response.job().report();
        System.out.println("Report Bucket: " + report.bucket());
        System.out.println("Report Prefix: " + report.prefix());
        System.out.println("Report Format: " + report.format());
        System.out.println("Report Enabled: " + report.enabled());
        System.out.println("Report Scope: " +
report.reportScopeAsString());
    })
    .exceptionally(ex -> {
        System.err.println("Failed to describe job: " + ex.getMessage());
        throw new RuntimeException(ex);
    });
```

```

}

/**
 * Creates an asynchronous S3 job using the AWS Java SDK.
 *
 * @param accountId      the AWS account ID associated with the job
 * @param iamRoleArn      the ARN of the IAM role to be used for the job
 * @param manifestLocation the location of the job manifest file in S3
 * @param reportBucketName the name of the S3 bucket to store the job report
 * @param uuid            a unique identifier for the job
 * @return a CompletableFuture that represents the asynchronous creation of
the S3 job.
 *
 *      The CompletableFuture will return the job ID if the job is created
successfully,
 *
 *      or throw an exception if there is an error.
 */
public CompletableFuture<String> createS3JobAsync(String accountId, String
iamRoleArn,
                                                    String manifestLocation,
String reportBucketName, String uuid) {

    String[] bucketName = new String[]{" "};
    String[] parts = reportBucketName.split("::");
    if (parts.length > 1) {
        bucketName[0] = parts[1];
    } else {
        System.out.println("The input string does not contain the expected
format.");
    }

    return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
        .thenCompose(eTag -> {
            ArrayList<S3Tag> tagSet = new ArrayList<>();
            S3Tag s3Tag = S3Tag.builder()
                .key("keyOne")
                .value("ValueOne")
                .build();
            S3Tag s3Tag2 = S3Tag.builder()
                .key("keyTwo")
                .value("ValueTwo")
                .build();
            tagSet.add(s3Tag);
            tagSet.add(s3Tag2);
        });
}

```

```
S3SetObjectTaggingOperation objectTaggingOperation =
S3SetObjectTaggingOperation.builder()
    .tagSet(tagSet)
    .build();

JobOperation jobOperation = JobOperation.builder()
    .s3PutObjectTagging(objectTaggingOperation)
    .build();

JobManifestLocation jobManifestLocation =
JobManifestLocation.builder()
    .objectArn(manifestLocation)
    .eTag(eTag)
    .build();

JobManifestSpec manifestSpec = JobManifestSpec.builder()
    .fieldsWithStrings("Bucket", "Key")
    .format("S3BatchOperations_CSV_20180820")
    .build();

JobManifest jobManifest = JobManifest.builder()
    .spec(manifestSpec)
    .location(jobManifestLocation)
    .build();

JobReport jobReport = JobReport.builder()
    .bucket(reportBucketName)
    .prefix("reports")
    .format("Report_CSV_20180820")
    .enabled(true)
    .reportScope("AllTasks")
    .build();

CreateJobRequest jobRequest = CreateJobRequest.builder()
    .accountId(accountId)
    .description("Job created using the AWS Java SDK")
    .manifest(jobManifest)
    .operation(jobOperation)
    .report(jobReport)
    .priority(42)
    .roleArn(iamRoleArn)
    .clientRequestToken(uuid)
    .confirmationRequired(false)
```

```

        .build();

        // Create the job asynchronously.
        return getAsyncClient().createJob(jobRequest)
            .thenApply(CreateJobResponse::jobId);
    })
    .handle((jobId, ex) -> {
        if (ex != null) {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof S3ControlException) {
                throw new CompletionException(cause);
            } else {
                throw new RuntimeException(cause);
            }
        }
        return jobId;
    });
}

/**
 * Retrieves the ETag (Entity Tag) for an object stored in an Amazon S3
bucket.
 *
 * @param bucketName the name of the Amazon S3 bucket where the object is
stored
 * @param key the key (file name) of the object in the Amazon S3 bucket
 * @return the ETag of the object
 */
public String getETag(String bucketName, String key) {
    S3Client s3Client = S3Client.builder()
        .region(Region.US_EAST_1)
        .build();

    HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
    return headObjectResponse.eTag();
}

```

```
/**
 * Asynchronously adds tags to a job in the system.
 *
 * @param jobId      the ID of the job to add tags to
 * @param accountId  the account ID associated with the job
 * @return a CompletableFuture that completes when the tagging operation is
finished
 */
public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
    S3Tag departmentTag = S3Tag.builder()
        .key("department")
        .value("Marketing")
        .build();

    S3Tag fiscalYearTag = S3Tag.builder()
        .key("FiscalYear")
        .value("2020")
        .build();

    PutJobTaggingRequest putJobTaggingRequest =
PutJobTaggingRequest.builder()
        .jobId(jobId)
        .accountId(accountId)
        .tags(departmentTag, fiscalYearTag)
        .build();

    return asyncClient.putJobTagging(putJobTaggingRequest)
        .thenRun(() -> {
            System.out.println("Additional Tags were added to job " + jobId);
        })
        .exceptionally(ex -> {
            System.err.println("Failed to add tags to job: " +
ex.getMessage());
            throw new RuntimeException(ex); // Propagate the exception
        });
}

// Setup the S3 bucket required for this scenario.
/**
 * Creates an Amazon S3 bucket with the specified name.
 *
 * @param bucketName the name of the S3 bucket to create
 * @throws S3Exception if there is an error creating the bucket
 */
```

```
    */
    public void createBucket(String bucketName) {
        try {
            S3Client s3Client = S3Client.builder()
                .region(Region.US_EAST_1)
                .build();

            S3Waiter s3Waiter = s3Client.waiter();
            CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
                .bucket(bucketName)
                .build();

            s3Client.createBucket(bucketRequest);
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();

            // Wait until the bucket is created and print out the response.
            WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println(bucketName + " is ready");

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Uploads a file to an Amazon S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param fileName the name of the file to be uploaded
     * @throws RuntimeException if an error occurs during the file upload
     */
    public void populateBucket(String bucketName, String fileName) {
        // Define the path to the directory.
        Path filePath = Paths.get("src/main/resources/batch/",
fileName).toAbsolutePath();
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(fileName)
```

```
        .build());

        CompletableFuture<PutObjectResponse> future =
getS3AsyncClient().putObject(putOb, AsyncRequestBody.fromFile(filePath));
        future.whenComplete((result, ex) -> {
            if (ex != null) {
                System.err.println("Error uploading file: " + ex.getMessage());
            } else {
                System.out.println("Successfully placed " + fileName + " into
bucket " + bucketName);
            }
        }).join();
    }

    // Update the bucketName in CSV.
    public void updateCSV(String newValue) {
        Path csvFilePath = Paths.get("src/main/resources/batch/job-
manifest.csv").toAbsolutePath();
        try {
            // Read all lines from the CSV file.
            List<String> lines = Files.readAllLines(csvFilePath);

            // Update the first value in each line.
            List<String> updatedLines = lines.stream()
                .map(line -> {
                    String[] parts = line.split(",");
                    parts[0] = newValue;
                    return String.join(",", parts);
                })
                .collect(Collectors.toList());

            // Write the updated lines back to the CSV file
            Files.write(csvFilePath, updatedLines);
            System.out.println("CSV file updated successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /**
     * Deletes an object from an Amazon S3 bucket asynchronously.
     *
     * @param bucketName The name of the S3 bucket where the object is stored.
```

```

    * @param objectName The name of the object to be deleted.
    * @return A {@link CompletableFuture} that completes when the object has
    been deleted,
    *         or throws a {@link RuntimeException} if an error occurs during the
    deletion.
    */
    public CompletableFuture<Void> deleteBucketObjects(String bucketName, String
objectName) {
        ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
        toDelete.add(ObjectIdentifier.builder()
            .key(objectName)
            .build());

        DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(Delete.builder()
                .objects(toDelete).build())
            .build();

        return getS3AsyncClient().deleteObjects(dor)
            .thenAccept(result -> {
                System.out.println("The object was deleted!");
            })
            .exceptionally(ex -> {
                throw new RuntimeException("Error deleting object: " +
ex.getMessage(), ex);
            });
    }

    /**
    * Deletes a folder and all its contents asynchronously from an Amazon S3
    bucket.
    *
    * @param bucketName the name of the S3 bucket containing the folder to be
    deleted
    * @return a {@link CompletableFuture} that completes when the folder and its
    contents have been deleted
    * @throws RuntimeException if any error occurs during the deletion process
    */
    public void deleteBucketFolderAsync(String bucketName) {
        String folderName = "reports/";
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .prefix(folderName)

```



```

        .build();

        CompletableFuture<ListObjectsV2Response> listObjectsFuture =
getS3AsyncClient().listObjectsV2(request);
        listObjectsFuture.thenCompose(response -> {
            List<CompletableFuture<DeleteObjectResponse>> deleteFutures =
response.contents().stream()
                .map(obj -> {
                    DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                        .bucket(bucketName)
                        .key(obj.key())
                        .build();
                    return getS3AsyncClient().deleteObject(deleteRequest)
                        .thenApply(deleteResponse -> {
                            System.out.println("Deleted object: " + obj.key());
                            return deleteResponse;
                        });
                })
                .collect(Collectors.toList());

            return CompletableFuture.allOf(deleteFutures.toArray(new
CompletableFuture[0]))
                .thenCompose(v -> {
                    // Delete the folder.
                    DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                        .bucket(bucketName)
                        .key(folderName)
                        .build();
                    return getS3AsyncClient().deleteObject(deleteRequest)
                        .thenApply(deleteResponse -> {
                            System.out.println("Deleted folder: " + folderName);
                            return deleteResponse;
                        });
                });
        }).join();
    }

/**
 * Deletes an Amazon S3 bucket.
 *
 * @param bucketName the name of the bucket to delete

```

```
    * @return a {@link CompletableFuture} that completes when the bucket has
    been deleted, or exceptionally if there is an error
    * @throws RuntimeException if there is an error deleting the bucket
    */
    public CompletableFuture<Void> deleteBucket(String bucketName) {
        S3AsyncClient s3Client = getS3AsyncClient();
        return s3Client.deleteBucket(DeleteBucketRequest.builder()
            .bucket(bucketName)
            .build())
            .thenAccept(deleteBucketResponse -> {
                System.out.println(bucketName + " was deleted");
            })
            .exceptionally(ex -> {
                // Handle the exception or rethrow it.
                throw new RuntimeException("Failed to delete bucket: " +
bucketName, ex);
            });
    }

    /**
     * Uploads a set of files to an Amazon S3 bucket.
     *
     * @param bucketName the name of the S3 bucket to upload the files to
     * @param fileNames an array of file names to be uploaded
     * @param actions an instance of {@link S3BatchActions} that provides the
    implementation for the necessary S3 operations
     * @throws IOException if there's an error creating the text files or
    uploading the files to the S3 bucket
     */
    public static void uploadFilesToBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
        actions.updateCSV(bucketName);
        createTextFiles(fileNames);
        for (String fileName : fileNames) {
            actions.populateBucket(bucketName, fileName);
        }
        System.out.println("All files are placed in the S3 bucket " +
bucketName);
    }

    /**
     * Deletes the specified files from the given S3 bucket.
     *
     * @param bucketName the name of the S3 bucket
```

```
* @param fileNames an array of file names to be deleted from the bucket
* @param actions the S3BatchActions instance to be used for the file
deletion
* @throws IOException if an I/O error occurs during the file deletion
*/
public void deleteFilesFromBucket(String bucketName, String[] fileNames,
S3BatchActions actions) throws IOException {
    for (String fileName : fileNames) {
        actions.deleteBucketObjects(bucketName, fileName)
            .thenRun(() -> System.out.println("Object deletion completed"))
            .exceptionally(ex -> {
                System.err.println("Error occurred: " + ex.getMessage());
                return null;
            });
    }
    System.out.println("All files have been deleted from the bucket " +
bucketName);
}

public static void createTextFiles(String[] fileNames) {
    String currentDirectory = System.getProperty("user.dir");
    String directoryPath = currentDirectory + "\\src\\main\\resources\\
\\batch";
    Path path = Paths.get(directoryPath);

    try {
        // Create the directory if it doesn't exist.
        if (Files.notExists(path)) {
            Files.createDirectories(path);
            System.out.println("Created directory: " + path.toString());
        } else {
            System.out.println("Directory already exists: " +
path.toString());
        }

        for (String fileName : fileNames) {
            // Check if the file is a .txt file.
            if (fileName.endsWith(".txt")) {
                // Define the path for the new file.
                Path filePath = path.resolve(fileName);
                System.out.println("Attempting to create file: " +
filePath.toString());

                // Create and write content to the new file.
```

```
        Files.write(filePath, "This is a test".getBytes());

        // Verify the file was created.
        if (Files.exists(filePath)) {
            System.out.println("Successfully created file: " +
filePath.toString());
        } else {
            System.out.println("Failed to create file: " +
filePath.toString());
        }
    }

    } catch (IOException e) {
        System.err.println("An error occurred: " + e.getMessage());
        e.printStackTrace();
    }
}

public String getAccountId() {
    StsClient stsClient = StsClient.builder()
        .region(Region.US_EAST_1)
        .build();

    GetCallerIdentityResponse callerIdentityResponse =
stsClient.getCallerIdentity();
    return callerIdentityResponse.account();
}
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK for Java 2.x .
 - [CreateJob](#)
 - [DeleteJobTagging](#)
 - [DescribeJob](#)
 - [GetJobTagging](#)
 - [ListJobs](#)
 - [PutJobTagging](#)
 - [UpdateJobPriority](#)

- [UpdateJobStatus](#)

Python

SDK per Python (Boto3)

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Scopri lo scenario di base di S3 Batch.

```
class S3BatchWrapper:
    """Wrapper class for managing S3 Batch Operations."""

    def __init__(self, s3_client: Any, s3control_client: Any, sts_client: Any) ->
None:
        """
        Initializes the S3BatchWrapper with AWS service clients.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                        access to AWS S3 services.
        :param s3control_client: A Boto3 Amazon S3 Control client. This client
provides
                        low-level access to AWS S3 Control services.
        :param sts_client: A Boto3 AWS STS client. This client provides low-level
                        access to AWS STS services.
        """
        self.s3_client = s3_client
        self.s3control_client = s3control_client
        self.sts_client = sts_client
        # Get region from the client for bucket creation logic
        self.region_name = self.s3_client.meta.region_name

    def get_account_id(self) -> str:
        """
        Get AWS account ID.

        Returns:
```

```

        str: AWS account ID
    """
    return self.sts_client.get_caller_identity()["Account"]

def create_bucket(self, bucket_name: str) -> None:
    """
    Create an S3 bucket.

    Args:
        bucket_name (str): Name of the bucket to create

    Raises:
        ClientError: If bucket creation fails
    """
    try:
        if self.region_name and self.region_name != 'us-east-1':
            self.s3_client.create_bucket(
                Bucket=bucket_name,
                CreateBucketConfiguration={
                    'LocationConstraint': self.region_name
                }
            )
        else:
            self.s3_client.create_bucket(Bucket=bucket_name)
            print(f"Created bucket: {bucket_name}")
    except ClientError as e:
        print(f"Error creating bucket: {e}")
        raise

def upload_files_to_bucket(self, bucket_name: str, file_names: List[str]) -> str:
    """
    Upload files to S3 bucket including manifest file.

    Args:
        bucket_name (str): Target bucket name
        file_names (list): List of file names to upload

    Returns:
        str: ETag of the manifest file

    Raises:
        ClientError: If file upload fails
    """

```

```

try:
    for file_name in file_names:
        if file_name != "job-manifest.csv":
            content = f"Content for {file_name}"
            self.s3_client.put_object(
                Bucket=bucket_name,
                Key=file_name,
                Body=content.encode('utf-8')
            )
            print(f"Uploaded {file_name} to {bucket_name}")

    manifest_content = ""
    for file_name in file_names:
        if file_name != "job-manifest.csv":
            manifest_content += f"{bucket_name},{file_name}\n"

    manifest_response = self.s3_client.put_object(
        Bucket=bucket_name,
        Key="job-manifest.csv",
        Body=manifest_content.encode('utf-8')
    )
    print(f"Uploaded manifest file to {bucket_name}")
    print(f"Manifest content:\n{manifest_content}")
    return manifest_response['ETag'].strip('')

except ClientError as e:
    print(f"Error uploading files: {e}")
    raise

def create_s3_batch_job(self, account_id: str, role_arn: str,
    manifest_location: str,
    report_bucket_name: str) -> str:
    """
    Create an S3 batch operation job.

    Args:
        account_id (str): AWS account ID
        role_arn (str): IAM role ARN for batch operations
        manifest_location (str): Location of the manifest file
        report_bucket_name (str): Bucket for job reports

    Returns:
        str: Job ID

```

```

Raises:
    ClientError: If job creation fails
"""
try:
    bucket_name = manifest_location.split(':::')[1].split('/')[0]
    manifest_key = 'job-manifest.csv'
    manifest_obj = self.s3_client.head_object(
        Bucket=bucket_name,
        Key=manifest_key
    )
    etag = manifest_obj['ETag'].strip('""')

    response = self.s3control_client.create_job(
        AccountId=account_id,
        Operation={
            'S3PutObjectTagging': {
                'TagSet': [
                    {
                        'Key': 'BatchTag',
                        'Value': 'BatchValue'
                    },
                ],
            }
        },
        Report={
            'Bucket': report_bucket_name,
            'Format': 'Report_CSV_20180820',
            'Enabled': True,
            'Prefix': 'batch-op-reports',
            'ReportScope': 'AllTasks'
        },
        Manifest={
            'Spec': {
                'Format': 'S3BatchOperations_CSV_20180820',
                'Fields': ['Bucket', 'Key']
            },
            'Location': {
                'ObjectArn': manifest_location,
                'ETag': etag
            }
        },
        Priority=10,
        RoleArn=role_arn,
        Description='Batch job for tagging objects',

```



```

        ConfirmationRequired=True
    )
    job_id = response['JobId']
    print(f"The Job id is {job_id}")
    return job_id
except ClientError as e:
    print(f"Error creating batch job: {e}")
    if 'Message' in str(e):
        print(f"Detailed error message: {e.response['Message']}")
    raise

def check_job_failure_reasons(self, job_id: str, account_id: str) ->
List[Dict[str, Any]]:
    """
    Check for any failure reasons of a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID

    Returns:
        list: List of failure reasons

    Raises:
        ClientError: If checking job failure reasons fails
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        if 'FailureReasons' in response['Job']:
            for reason in response['Job']['FailureReasons']:
                print(f"- {reason}")
            return response['Job'].get('FailureReasons', [])
    except ClientError as e:
        print(f"Error checking job failure reasons: {e}")
        raise

def wait_for_job_ready(self, job_id: str, account_id: str, desired_status:
str = 'Ready') -> bool:
    """
    Wait for a job to reach the desired status.

```

```

Args:
    job_id (str): ID of the batch job
    account_id (str): AWS account ID
    desired_status (str): Target status to wait for

Returns:
    bool: True if desired status is reached, False otherwise

Raises:
    ClientError: If checking job status fails
"""
print(f"Waiting for job to become {desired_status}...")
max_attempts = 60
attempt = 0
while attempt < max_attempts:
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")
        if current_status == desired_status:
            return True
        if current_status == 'Suspended':
            print("Job is in Suspended state, can proceed with
activation")
            return True
        if current_status in ['Active', 'Failed', 'Cancelled',
'Complete']:
            print(f"Job is in {current_status} state, cannot reach
{desired_status} status")
            if 'FailureReasons' in response['Job']:
                print("Failure reasons:")
                for reason in response['Job']['FailureReasons']:
                    print(f"- {reason}")
            return False

        time.sleep(20)
        attempt += 1
    except ClientError as e:
        print(f"Error checking job status: {e}")
        raise
print(f"Timeout waiting for job to become {desired_status}")

```

```

        return False

def update_job_priority(self, job_id: str, account_id: str) -> None:
    """
    Update the priority of a batch job and start it.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")

        if current_status in ['Ready', 'Suspended']:
            self.s3control_client.update_job_priority(
                AccountId=account_id,
                JobId=job_id,
                Priority=60
            )
            print("The job priority was updated")

            try:
                self.s3control_client.update_job_status(
                    AccountId=account_id,
                    JobId=job_id,
                    RequestedJobStatus='Ready'
                )
                print("Job activated successfully")
            except ClientError as activation_error:
                print(f"Note: Could not activate job automatically: {activation_error}")
                print("Job priority was updated successfully. Job may need manual activation in the console.")
            elif current_status in ['Active', 'Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - priority cannot be updated")

            if current_status == 'Completing':
                print("Job is finishing up and will complete soon.")
            elif current_status == 'Complete':

```

```

        print("Job has already completed successfully.")
    else:
        print("Job is currently running.")
    else:
        print(f"Job is in '{current_status}' state - priority update not
allowed")

except ClientError as e:
    print(f"Error updating job priority: {e}")
    print("Continuing with the scenario...")
    return

def cancel_job(self, job_id: str, account_id: str) -> None:
    """
    Cancel an S3 batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")

        if current_status in ['Ready', 'Suspended', 'Active']:
            self.s3control_client.update_job_status(
                AccountId=account_id,
                JobId=job_id,
                RequestedJobStatus='Cancelled'
            )
            print(f"Job {job_id} was successfully canceled.")
        elif current_status in ['Completing', 'Complete']:
            print(f"Job is in '{current_status}' state - cannot be
cancelled")

            if current_status == 'Completing':
                print("Job is finishing up and will complete soon.")
            elif current_status == 'Complete':
                print("Job has already completed successfully.")
        else:
            print(f"Job is in '{current_status}' state - cancel not allowed")

```

```

        except ClientError as e:
            print(f"Error canceling job: {e}")
            raise

def describe_job_details(self, job_id: str, account_id: str) -> None:
    """
    Describe detailed information about a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        job = response['Job']
        print(f"Job ID: {job['JobId']}")
        print(f"Description: {job.get('Description', 'N/A')}")
        print(f"Status: {job['Status']}")
        print(f"Role ARN: {job['RoleArn']}")
        print(f"Priority: {job['Priority']}")
        if 'ProgressSummary' in job:
            progress = job['ProgressSummary']
            print(f"Progress Summary:
Total={progress.get('TotalNumberOfTasks', 0)}, "
                  f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
                  f"Failed={progress.get('NumberOfTasksFailed', 0)}")
    except ClientError as e:
        print(f"Error describing job: {e}")
        raise

def get_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Get tags associated with a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.get_job_tagging(
            AccountId=account_id,

```

```

        JobId=job_id
    )
    tags = response.get('Tags', [])
    if tags:
        print(f"Tags for job {job_id}:")
        for tag in tags:
            print(f"  {tag['Key']}: {tag['Value']}")
    else:
        print(f"No tags found for job ID: {job_id}")
except ClientError as e:
    print(f"Error getting job tags: {e}")
    raise

def put_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Add tags to a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.put_job_tagging(
            AccountId=account_id,
            JobId=job_id,
            Tags=[
                {'Key': 'Environment', 'Value': 'Development'},
                {'Key': 'Team', 'Value': 'DataProcessing'}
            ]
        )
        print(f"Additional tags were added to job {job_id}")
    except ClientError as e:
        print(f"Error adding job tags: {e}")
        raise

def list_jobs(self, account_id: str) -> None:
    """
    List all batch jobs for the account.

    Args:
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.list_jobs(

```

```

        AccountId=account_id,
        JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']
    )
    jobs = response.get('Jobs', [])
    for job in jobs:
        print(f"The job id is {job['JobId']}")
        print(f"The job priority is {job['Priority']}")
except ClientError as e:
    print(f"Error listing jobs: {e}")
    raise

def delete_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Delete all tags from a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.delete_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        print(f"You have successfully deleted {job_id} tagging.")
    except ClientError as e:
        print(f"Error deleting job tags: {e}")
        raise

def cleanup_resources(self, bucket_name: str, file_names: List[str]) -> None:
    """
    Clean up all resources created during the scenario.

    Args:
        bucket_name (str): Name of the bucket to clean up
        file_names (list): List of files to delete

    Raises:
        ClientError: If cleanup fails
    """
    try:
        for file_name in file_names:
            self.s3_client.delete_object(Bucket=bucket_name, Key=file_name)

```

```
        print(f"Deleted {file_name}")

    response = self.s3_client.list_objects_v2(
        Bucket=bucket_name,
        Prefix='batch-op-reports/'
    )
    if 'Contents' in response:
        for obj in response['Contents']:
            self.s3_client.delete_object(
                Bucket=bucket_name,
                Key=obj['Key']
            )
            print(f"Deleted {obj['Key']}")

    self.s3_client.delete_bucket(Bucket=bucket_name)
    print(f"Deleted bucket {bucket_name}")
except ClientError as e:
    print(f"Error in cleanup: {e}")
    raise
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Python (Boto3).
 - [CreateJob](#)
 - [DeleteJobTagging](#)
 - [DescribeJob](#)
 - [GetJobTagging](#)
 - [ListJobs](#)
 - [PutJobTagging](#)
 - [UpdateJobPriority](#)
 - [UpdateJobStatus](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Azioni per Amazon S3 Control utilizzando AWS SDK

I seguenti esempi di codice mostrano come eseguire singole azioni di controllo di Amazon S3 con AWS gli SDK. Ogni esempio include un collegamento a GitHub, dove puoi trovare le istruzioni per la configurazione e l'esecuzione del codice.

Gli esempi seguenti includono solo le azioni più comunemente utilizzate. Per un elenco completo, consulta la [documentazione di riferimento dell'API Amazon S3 Control](#).

Esempi

- [Da utilizzare CreateJobcon un AWS SDK o CLI](#)
- [Utilizzare DeleteJobTaggingcon un AWS SDK](#)
- [Utilizzare DescribeJobcon un AWS SDK o CLI](#)
- [Utilizzare GetJobTaggingcon un AWS SDK](#)
- [Utilizzare PutJobTaggingcon un AWS SDK](#)
- [Utilizzare UpdateJobPrioritycon un AWS SDK o CLI](#)
- [Utilizzare UpdateJobStatuscon un AWS SDK o CLI](#)

Da utilizzare **CreateJob**con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare CreateJob.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Informazioni di base](#)

CLI

AWS CLI

Come creare un processo Operazioni in batch Amazon S3

L'esempio create-job seguente crea un processo di operazioni in batch Amazon S3 con cui taggare agli oggetti come `confidential`` in the bucket ``employee-records.

```
aws s3control create-job \  
  --account-id 123456789012 \
```

```
--operation '{"S3PutObjectTagging": { "TagSet": [{"Key":"confidential",
"Value":"true"}] }}' \
--report '{"Bucket":"arn:aws:s3:::employee-records-logs", "Prefix":"batch-op-
create-job",
"Format":"Report_CSV_20180820", "Enabled":true, "ReportScope":"AllTasks"}' \
--manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820", "Fields":
[{"Bucket", "Key"}], "Location":{"ObjectArn":"arn:aws:s3:::employee-records-logs/
inv-report/7a6a9be4-072c-407e-85a2-
ec3e982f773e.csv", "ETag":"69f52a4e9f797e987155d9c8f5880897"}}' \
--priority 42 \
--role-arn arn:aws:iam::123456789012:role/S3BatchJobRole
```

Output:

```
{
  "JobId": "93735294-df46-44d5-8638-6356f335324e"
}
```

- Per i dettagli sull'API, consulta [CreateJob AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea un processo S3 asincrono.

```
/**
 * Creates an asynchronous S3 job using the AWS Java SDK.
 *
 * @param accountId      the AWS account ID associated with the job
 * @param iamRoleArn     the ARN of the IAM role to be used for the job
 * @param manifestLocation the location of the job manifest file in S3
 * @param reportBucketName the name of the S3 bucket to store the job report
 * @param uuid           a unique identifier for the job
```

```
* @return a CompletableFuture that represents the asynchronous creation of
the S3 job.
*       The CompletableFuture will return the job ID if the job is created
successfully,
*       or throw an exception if there is an error.
*/
public CompletableFuture<String> createS3JobAsync(String accountId, String
iamRoleArn,
                                                String manifestLocation,
String reportBucketName, String uuid) {

    String[] bucketName = new String[]{" "};
    String[] parts = reportBucketName.split("::");
    if (parts.length > 1) {
        bucketName[0] = parts[1];
    } else {
        System.out.println("The input string does not contain the expected
format.");
    }

    return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
        .thenCompose(eTag -> {
            ArrayList<S3Tag> tagSet = new ArrayList<>();
            S3Tag s3Tag = S3Tag.builder()
                .key("keyOne")
                .value("ValueOne")
                .build();
            S3Tag s3Tag2 = S3Tag.builder()
                .key("keyTwo")
                .value("ValueTwo")
                .build();
            tagSet.add(s3Tag);
            tagSet.add(s3Tag2);

            S3SetObjectTaggingOperation objectTaggingOperation =
S3SetObjectTaggingOperation.builder()
                .tagSet(tagSet)
                .build();

            JobOperation jobOperation = JobOperation.builder()
                .s3PutObjectTagging(objectTaggingOperation)
                .build();
```

```
JobManifestLocation jobManifestLocation =
JobManifestLocation.builder()
    .objectArn(manifestLocation)
    .eTag(eTag)
    .build();

JobManifestSpec manifestSpec = JobManifestSpec.builder()
    .fieldsWithStrings("Bucket", "Key")
    .format("S3BatchOperations_CSV_20180820")
    .build();

JobManifest jobManifest = JobManifest.builder()
    .spec(manifestSpec)
    .location(jobManifestLocation)
    .build();

JobReport jobReport = JobReport.builder()
    .bucket(reportBucketName)
    .prefix("reports")
    .format("Report_CSV_20180820")
    .enabled(true)
    .reportScope("AllTasks")
    .build();

CreateJobRequest jobRequest = CreateJobRequest.builder()
    .accountId(accountId)
    .description("Job created using the AWS Java SDK")
    .manifest(jobManifest)
    .operation(jobOperation)
    .report(jobReport)
    .priority(42)
    .roleArn(iamRoleArn)
    .clientRequestToken(uuid)
    .confirmationRequired(false)
    .build();

// Create the job asynchronously.
return getAsyncClient().createJob(jobRequest)
    .thenApply(CreateJobResponse::jobId);
})
.handle((jobId, ex) -> {
    if (ex != null) {
        Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
```

```

        if (cause instanceof S3ControlException) {
            throw new CompletionException(cause);
        } else {
            throw new RuntimeException(cause);
        }
    }
    return jobId;
});
}

```

Crea un processo di conservazione della conformità.

```

/**
 * Creates a compliance retention job in Amazon S3 Control.
 * <p>
 * A compliance retention job in Amazon S3 Control is a feature that allows
you to
 * set a retention period for objects stored in an S3 bucket.
 * This feature is particularly useful for organizations that need to comply
with
 * regulatory requirements or internal policies that mandate the retention of
data for
 * a specific duration.
 *
 * @param s3ControlClient The S3ControlClient instance to use for the API
call.
 * @return The job ID of the created compliance retention job.
 */
public static String createComplianceRetentionJob(final S3ControlClient
s3ControlClient, String roleArn, String bucketName, String accountId) {
    final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-
bucket/compliance-objects-manifest.csv";
    final String manifestObjectVersionId = "your-object-version-Id";

    Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
    JobOperation jobOperation = JobOperation.builder()
        .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
            .retention(S3Retention.builder()
                .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                .retainUntilDate(jan2025)
                .build())
        .build())

```

```
        .build())
    .build();

JobManifestLocation manifestLocation = JobManifestLocation.builder()
    .objectArn(manifestObjectArn)
    .eTag(manifestObjectVersionId)
    .build();

JobManifestSpec manifestSpec = JobManifestSpec.builder()
    .fieldsWithStrings("Bucket", "Key")
    .format("S3BatchOperations_CSV_20180820")
    .build();

JobManifest manifestToPublicApi = JobManifest.builder()
    .location(manifestLocation)
    .spec(manifestSpec)
    .build();

// Report details.
final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
final String jobReportPrefix = "reports/compliance-objects-bops";

JobReport jobReport = JobReport.builder()
    .enabled(true)
    .reportScope(JobReportScope.ALL_TASKS)
    .bucket(jobReportBucketArn)
    .prefix(jobReportPrefix)
    .format(JobReportFormat.REPORT_CSV_20180820)
    .build();

final Boolean requiresConfirmation = true;
final int priority = 10;
CreateJobRequest request = CreateJobRequest.builder()
    .accountId(accountId)
    .description("Set compliance retain-until to 1 Jan 2025")
    .manifest(manifestToPublicApi)
    .operation(jobOperation)
    .priority(priority)
    .roleArn(roleArn)
    .report(jobReport)
    .confirmationRequired(requiresConfirmation)
    .build();

// Create the job and get the result.
```

```

        CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }

```

Crea un processo di conservazione a fini legali.

```

/**
 * Creates a compliance retention job in Amazon S3 Control.
 * <p>
 * A compliance retention job in Amazon S3 Control is a feature that allows
you to
 * set a retention period for objects stored in an S3 bucket.
 * This feature is particularly useful for organizations that need to comply
with
 * regulatory requirements or internal policies that mandate the retention of
data for
 * a specific duration.
 *
 * @param s3ControlClient The S3ControlClient instance to use for the API
call.
 * @return The job ID of the created compliance retention job.
 */
public static String createComplianceRetentionJob(final S3ControlClient
s3ControlClient, String roleArn, String bucketName, String accountId) {
    final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-
bucket/compliance-objects-manifest.csv";
    final String manifestObjectVersionId = "your-object-version-Id";

    Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
    JobOperation jobOperation = JobOperation.builder()
        .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
            .retention(S3Retention.builder()
                .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                .retainUntilDate(jan2025)
                .build())
            .build())
        .build();

    JobManifestLocation manifestLocation = JobManifestLocation.builder()
        .objectArn(manifestObjectArn)
        .eTag(manifestObjectVersionId)

```

```
        .build();

    JobManifestSpec manifestSpec = JobManifestSpec.builder()
        .fieldsWithStrings("Bucket", "Key")
        .format("S3BatchOperations_CSV_20180820")
        .build();

    JobManifest manifestToPublicApi = JobManifest.builder()
        .location(manifestLocation)
        .spec(manifestSpec)
        .build();

    // Report details.
    final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
    final String jobReportPrefix = "reports/compliance-objects-bops";

    JobReport jobReport = JobReport.builder()
        .enabled(true)
        .reportScope(JobReportScope.ALL_TASKS)
        .bucket(jobReportBucketArn)
        .prefix(jobReportPrefix)
        .format(JobReportFormat.REPORT_CSV_20180820)
        .build();

    final Boolean requiresConfirmation = true;
    final int priority = 10;
    CreateJobRequest request = CreateJobRequest.builder()
        .accountId(accountId)
        .description("Set compliance retain-until to 1 Jan 2025")
        .manifest(manifestToPublicApi)
        .operation(jobOperation)
        .priority(priority)
        .roleArn(roleArn)
        .report(jobReport)
        .confirmationRequired(requiresConfirmation)
        .build();

    // Create the job and get the result.
    CreateJobResponse result = s3ControlClient.createJob(request);
    return result.jobId();
}
```


Crea un nuovo processo di conservazione della governance.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateGovernanceRetentionJob {

    public static void main(String[]args) throws ParseException {
        final String usage = ""

            Usage:
                <manifestObjectArn> <jobReportBucketArn> <roleArn> <accountId>
                <manifestObjectVersionId>

            Where:
                manifestObjectArn - The Amazon Resource Name (ARN) of the S3
                object that contains the manifest file for the governance objects.\s
                bucketName - The ARN of the S3 bucket where the job report will
                be stored.
                roleArn - The ARN of the IAM role that will be used to perform
                the governance retention operation.
                accountId - Your AWS account Id.
                manifestObjectVersionId = A unique value that is used as the
                `eTag` property of the `JobManifestLocation` object.
            """;

        if (args.length != 4) {
            System.out.println(usage);
            return;
        }

        String manifestObjectArn = args[0];
        String jobReportBucketArn = args[1];
        String roleArn = args[2];
        String accountId = args[3];
        String manifestObjectVersionId = args[4];

        S3ControlClient s3ControlClient = S3ControlClient.create();
```

```
        createGovernanceRetentionJob(s3ControlClient, manifestObjectArn,
        jobReportBucketArn, roleArn, accountId, manifestObjectVersionId);
    }

    public static String createGovernanceRetentionJob(final S3ControlClient
    s3ControlClient, String manifestObjectArn, String jobReportBucketArn, String
    roleArn, String accountId, String manifestObjectVersionId) throws ParseException
    {
        final JobManifestLocation manifestLocation =
        JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        final JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .format(JobManifestFormat.S3_BATCH_OPERATIONS_CSV_20180820)
            .fields(Arrays.asList(JobManifestFieldName.BUCKET,
        JobManifestFieldName.KEY))
            .build();

        final JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

        final String jobReportPrefix = "reports/governance-objects";
        final JobReport jobReport = JobReport.builder()
            .enabled(true)
            .reportScope(JobReportScope.ALL_TASKS)
            .bucket(jobReportBucketArn)
            .prefix(jobReportPrefix)
            .format(JobReportFormat.REPORT_CSV_20180820)
            .build();

        final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
        final Date jan30th = format.parse("30/01/2025");

        final S3SetObjectRetentionOperation s3SetObjectRetentionOperation =
        S3SetObjectRetentionOperation.builder()
            .retention(S3Retention.builder()
                .mode(S3ObjectLockRetentionMode.GOVERNANCE)
                .retainUntilDate(jan30th.toInstant())
                .build())
            .build();
    }
```

```

        final JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(s3SetObjectRetentionOperation)
            .build();

        final Boolean requiresConfirmation = true;
        final int priority = 10;

        final CreateJobRequest request = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Put governance retention")
            .manifest(manifestToPublicApi)
            .operation(jobOperation)
            .priority(priority)
            .roleArn(roleArn)
            .report(jobReport)
            .confirmationRequired(requiresConfirmation)
            .build();

        final CreateJobResponse result = s3ControlClient.createJob(request);
        return result.jobId();
    }
}

```

- Per i dettagli sull'API, consulta la [CreateJob](#) sezione AWS SDK for Java 2.x API Reference.

Python

SDK per Python (Boto3)

Note

C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

def create_s3_batch_job(self, account_id: str, role_arn: str,
    manifest_location: str,
    report_bucket_name: str) -> str:
    """
    Create an S3 batch operation job.

```

Args:

account_id (str): AWS account ID
 role_arn (str): IAM role ARN for batch operations
 manifest_location (str): Location of the manifest file
 report_bucket_name (str): Bucket for job reports

Returns:

str: Job ID

Raises:

ClientError: If job creation fails

"""

try:

```

bucket_name = manifest_location.split(':::')[1].split('/')[0]
manifest_key = 'job-manifest.csv'
manifest_obj = self.s3_client.head_object(
    Bucket=bucket_name,
    Key=manifest_key
)
etag = manifest_obj['ETag'].strip('')

```

```

response = self.s3control_client.create_job(
    AccountId=account_id,
    Operation={
        'S3PutObjectTagging': {
            'TagSet': [
                {
                    'Key': 'BatchTag',
                    'Value': 'BatchValue'
                },
            ],
        }
    },
    Report={
        'Bucket': report_bucket_name,
        'Format': 'Report_CSV_20180820',
        'Enabled': True,
        'Prefix': 'batch-op-reports',
        'ReportScope': 'AllTasks'
    },
    Manifest={
        'Spec': {
            'Format': 'S3BatchOperations_CSV_20180820',

```

```

        'Fields': ['Bucket', 'Key']
    },
    'Location': {
        'ObjectArn': manifest_location,
        'ETag': etag
    }
},
Priority=10,
RoleArn=role_arn,
Description='Batch job for tagging objects',
ConfirmationRequired=True
)
job_id = response['JobId']
print(f"The Job id is {job_id}")
return job_id
except ClientError as e:
    print(f"Error creating batch job: {e}")
    if 'Message' in str(e):
        print(f"Detailed error message: {e.response['Message']}")
    raise

```

- Per i dettagli sull'API, consulta [CreateJob AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

TRY.

```

" iv_manifest_arn  = 'arn:aws:s3::my-bucket/job-manifest.csv'
" iv_manifest_etag = 'abc123def456'
" iv_report_bucket = 'arn:aws:s3::my-report-bucket'
DATA(lo_result) = lo_s3c->createjob(
    iv_accountid      = iv_account_id
    iv_rolearn        = iv_role_arn
    iv_confirmationrequired = abap_true

```

```

        iv_priority          = 10
        iv_description       = 'Batch job for tagging objects'
        io_operation         = NEW /aws1/cl_s3cjoboperation(
            io_s3putobjecttagging = NEW /aws1/cl_s3cs3setobjecttagop(
                it_tagset = VALUE /aws1/cl_s3cs3tag=>tt_s3tagset(
                    ( NEW /aws1/cl_s3cs3tag(
                        iv_key   = 'BatchTag'
                        iv_value = 'BatchValue' ) )
                )
            )
        )
        io_manifest          = NEW /aws1/cl_s3cjobmanifest(
            io_spec           = NEW /aws1/cl_s3cjobmanifestspec(
                iv_format      = 'S3BatchOperations_CSV_20180820'
                it_fields      = VALUE /aws1/
cl_s3cjobmanifestfield00=>tt_jobmanifestfieldlist(
                    ( NEW /aws1/cl_s3cjobmanifestfield00( 'Bucket' ) )
                    ( NEW /aws1/cl_s3cjobmanifestfield00( 'Key' ) )
                )
            )
            io_location       = NEW /aws1/cl_s3cjobmanifestloc(
                iv_objectarn   = iv_manifest_arn
                iv_etag        = iv_manifest_etag
            )
        )
        io_report            = NEW /aws1/cl_s3cjobreport(
            iv_bucket         = iv_report_bucket
            iv_format         = 'Report_CSV_20180820'
            iv_enabled        = abap_true
            iv_prefix         = 'batch-op-reports'
            iv_reportscope    = 'AllTasks'
        )
    ).
    ov_job_id = lo_result->get_jobid( ).
    MESSAGE |S3 Batch Operations job created: { ov_job_id }| TYPE 'I'.
    CATCH /aws1/cx_s3cbadrequestex INTO DATA(lo_ex_bad).
    MESSAGE lo_ex_bad->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_bad.
    CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_cli.
    CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).

```

```

MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
RAISE EXCEPTION TYPE /aws1/cx_rt_generic
EXPORTING previous = lo_ex_srv.
ENDTRY.

```

- Per i dettagli sulle API, [CreateJob](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteJobTagging** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare DeleteJobTagging.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Informazioni di base](#)

Java

SDK per Java 2.x

Note

C'è altro su [GitHub](#). Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

/**
 * Asynchronously deletes the tags associated with a specific batch job.
 *
 * @param jobId      The ID of the batch job whose tags should be deleted.
 * @param accountId The ID of the account associated with the batch job.
 * @return A CompletableFuture that completes when the job tags have been
 *         successfully deleted, or an exception is thrown if the deletion fails.
 */
public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {

```

```

DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
    .accountId(accountId)
    .jobId(jobId)
    .build();

return asyncClient.deleteJobTagging(jobTaggingRequest)
    .thenAccept(response -> {
        System.out.println("You have successfully deleted " + jobId + "
tagging.");
    })
    .exceptionally(ex -> {
        System.err.println("Failed to delete job tags: " +
ex.getMessage());
        throw new RuntimeException(ex);
    });
}

```

- Per i dettagli sull'API, consulta la [DeleteJobTagging](#) sezione AWS SDK for Java 2.x API Reference.

Python

SDK per Python (Boto3)

Note

C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

def delete_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Delete all tags from a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:

```



```

        self.s3control_client.delete_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        print(f"You have successfully deleted {job_id} tagging.")
    except ClientError as e:
        print(f"Error deleting job tags: {e}")
        raise

```

- Per i dettagli sull'API, consulta [DeleteJobTagging AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

TRY.
    lo_s3c->deletejobtagging(
        iv_accountid = iv_account_id
        iv_jobid      = iv_job_id
    ).
    MESSAGE |Tags deleted from job { iv_job_id }| TYPE 'I'.
CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_nf.
CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_cli.
CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_srv.

```

```
ENDTRY.
```

- Per i dettagli sulle API, [DeleteJobTagging](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DescribeJob** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare `DescribeJob`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Informazioni di base](#)

CLI

AWS CLI

Come descrivere un processo Operazioni in batch Amazon S3

L'esempio `describe-job` seguente fornisce i parametri di configurazione e lo stato per il processo di operazioni batch specificato.

```
aws s3control describe-job \  
  --account-id 123456789012 \  
  --job-id 93735294-df46-44d5-8638-6356f335324e
```

Output:

```
{  
  "Job": {  
    "TerminationDate": "2019-10-03T21:49:53.944Z",  
    "JobId": "93735294-df46-44d5-8638-6356f335324e",  
    "FailureReasons": [],  
    "Manifest": {  
      "Spec": {
```

```

        "Fields": [
            "Bucket",
            "Key"
        ],
        "Format": "S3BatchOperations_CSV_20180820"
    },
    "Location": {
        "ETag": "69f52a4e9f797e987155d9c8f5880897",
        "ObjectArn": "arn:aws:s3:::employee-records-logs/inv-report/7a6a9be4-072c-407e-85a2-ec3e982f773e.csv"
    }
},
"Operation": {
    "S3PutObjectTagging": {
        "TagSet": [
            {
                "Value": "true",
                "Key": "confidential"
            }
        ]
    }
},
"RoleArn": "arn:aws:iam::123456789012:role/S3BatchJobRole",
"ProgressSummary": {
    "TotalNumberOfTasks": 8,
    "NumberOfTasksFailed": 0,
    "NumberOfTasksSucceeded": 8
},
"Priority": 42,
"Report": {
    "ReportScope": "AllTasks",
    "Format": "Report_CSV_20180820",
    "Enabled": true,
    "Prefix": "batch-op-create-job",
    "Bucket": "arn:aws:s3:::employee-records-logs"
},
"JobArn": "arn:aws:s3:us-west-2:123456789012:job/93735294-df46-44d5-8638-6356f335324e",
"CreationTime": "2019-10-03T21:48:48.048Z",
"Status": "Complete"
}
}

```

- Per i dettagli sull'API, consulta [DescribeJob AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 * Asynchronously describes the specified job.
 *
 * @param jobId      the ID of the job to describe
 * @param accountId the ID of the AWS account associated with the job
 * @return a {@link CompletableFuture} that completes when the job
description is available
 * @throws RuntimeException if an error occurs while describing the job
 */
public CompletableFuture<Void> describeJobAsync(String jobId, String
accountId) {
    DescribeJobRequest jobRequest = DescribeJobRequest.builder()
        .jobId(jobId)
        .accountId(accountId)
        .build();

    return getAsyncClient().describeJob(jobRequest)
        .thenAccept(response -> {
            System.out.println("Job ID: " + response.job().jobId());
            System.out.println("Description: " +
response.job().description());
            System.out.println("Status: " + response.job().statusAsString());
            System.out.println("Role ARN: " + response.job().roleArn());
            System.out.println("Priority: " + response.job().priority());
            System.out.println("Progress Summary: " +
response.job().progressSummary());

            // Print out details about the job manifest.
            JobManifest manifest = response.job().manifest();
            System.out.println("Manifest Location: " +
manifest.location().objectArn());
        });
}
```

```

        System.out.println("Manifest ETag: " +
manifest.location().eTag());

        // Print out details about the job operation.
        JobOperation operation = response.job().operation();
        if (operation.s3PutObjectTagging() != null) {
            System.out.println("Operation: S3 Put Object Tagging");
            System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
        }

        // Print out details about the job report.
        JobReport report = response.job().report();
        System.out.println("Report Bucket: " + report.bucket());
        System.out.println("Report Prefix: " + report.prefix());
        System.out.println("Report Format: " + report.format());
        System.out.println("Report Enabled: " + report.enabled());
        System.out.println("Report Scope: " +
report.reportScopeAsString());
    })
    .exceptionally(ex -> {
        System.err.println("Failed to describe job: " + ex.getMessage());
        throw new RuntimeException(ex);
    });
}

```

- Per i dettagli sull'API, consulta la [DescribeJob](#) sezione AWS SDK for Java 2.x API Reference.

Python

SDK per Python (Boto3)

Note

C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
def describe_job_details(self, job_id: str, account_id: str) -> None:
```

```

"""
Describe detailed information about a batch job.

Args:
    job_id (str): ID of the batch job
    account_id (str): AWS account ID
"""
try:
    response = self.s3control_client.describe_job(
        AccountId=account_id,
        JobId=job_id
    )
    job = response['Job']
    print(f"Job ID: {job['JobId']}")
    print(f"Description: {job.get('Description', 'N/A')}")
    print(f"Status: {job['Status']}")
    print(f"Role ARN: {job['RoleArn']}")
    print(f"Priority: {job['Priority']}")
    if 'ProgressSummary' in job:
        progress = job['ProgressSummary']
        print(f"Progress Summary:
Total={progress.get('TotalNumberOfTasks', 0)}, "
              f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
              f"Failed={progress.get('NumberOfTasksFailed', 0)}")
except ClientError as e:
    print(f"Error describing job: {e}")
    raise

```

- Per i dettagli sull'API, consulta [DescribeJob AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

TRY.

```

        oo_result = lo_s3c->describejob(           " oo_result is returned for
testing purposes.
        iv_accountid = iv_account_id
        iv_jobid      = iv_job_id
    ).
    DATA(lo_job) = oo_result->get_job( ).
    DATA(lv_status) = lo_job->get_status( ).
    DATA(lv_priority) = lo_job->get_priority( ).
    DATA(lo_progress) = lo_job->get_progresssummary( ).
    IF lo_progress IS NOT INITIAL.
        MESSAGE |Job { iv_job_id }: status={ lv_status },
priority={ lv_priority }, | &&
                |total={ lo_progress->get_totalnumberoftasks( ) }, | &&
                |succeeded={ lo_progress->get_numberoftaskssucceeded( ) }, | &&
                |failed={ lo_progress->get_numberoftasksfailed( ) }| TYPE 'I'.
    ELSE.
        MESSAGE |Job { iv_job_id }: status={ lv_status },
priority={ lv_priority }| TYPE 'I'.
    ENDIF.
    CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_nf.
    CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_cli.
    CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_srv.
ENDTRY.

```

- Per i dettagli sulle API, [DescribeJob](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetJobTagging** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare GetJobTagging.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Informazioni di base](#)

Java

SDK per Java 2.x

 Note

C'è altro su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 * Asynchronously retrieves the tags associated with a specific job in an AWS
 * account.
 *
 * @param jobId      the ID of the job for which to retrieve the tags
 * @param accountId the ID of the AWS account associated with the job
 * @return a {@link CompletableFuture} that completes when the job tags have
 *         been retrieved, or with an exception if the operation fails
 * @throws RuntimeException if an error occurs while retrieving the job tags
 */
public CompletableFuture<Void> getJobTagsAsync(String jobId, String
accountId) {
    GetJobTaggingRequest request = GetJobTaggingRequest.builder()
        .jobId(jobId)
        .accountId(accountId)
        .build();

    return asyncClient.getJobTagging(request)
        .thenAccept(response -> {
            List<S3Tag> tags = response.tags();
            if (tags.isEmpty()) {
                System.out.println("No tags found for job ID: " + jobId);
            } else {
                for (S3Tag tag : tags) {
                    System.out.println("Tag key is: " + tag.key());
                    System.out.println("Tag value is: " + tag.value());
                }
            }
        });
}
```



```

        }
    }
})
.exceptionally(ex -> {
    System.err.println("Failed to get job tags: " + ex.getMessage());
    throw new RuntimeException(ex); // Propagate the exception
});
}

```

- Per i dettagli sull'API, consulta la [GetJobTagging](#) sezione AWS SDK for Java 2.x API Reference.

Python

SDK per Python (Boto3)

Note

C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

def get_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Get tags associated with a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.get_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        tags = response.get('Tags', [])
        if tags:
            print(f"Tags for job {job_id}:")
            for tag in tags:
                print(f"  {tag['Key']}: {tag['Value']}")
    
```

```

        else:
            print(f"No tags found for job ID: {job_id}")
    except ClientError as e:
        print(f"Error getting job tags: {e}")
        raise

```

- Per i dettagli sull'API, consulta [GetJobTagging AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

TRY.
    oo_result = lo_s3c->getjobtagging(          " oo_result is returned for
testing purposes.
    iv_accountid = iv_account_id
    iv_jobid      = iv_job_id
    ).
    DATA(lt_tags) = oo_result->get_tags( ).
    MESSAGE |Retrieved { lines( lt_tags ) } tag(s) for job { iv_job_id }|
TYPE 'I'.
    CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_nf.
    CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_cli.
    CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_srv.
ENDTRY.

```

- Per i dettagli sulle API, [GetJobTagging](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutJobTagging** con un AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare PutJobTagging.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Informazioni di base](#)

Java

SDK per Java 2.x

Note

C'è altro su [GitHub](#). Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 * Asynchronously adds tags to a job in the system.
 *
 * @param jobId      the ID of the job to add tags to
 * @param accountId the account ID associated with the job
 * @return a CompletableFuture that completes when the tagging operation is
 * finished
 */
public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
    S3Tag departmentTag = S3Tag.builder()
        .key("department")
        .value("Marketing")
```

```
        .build();

        S3Tag fiscalYearTag = S3Tag.builder()
            .key("FiscalYear")
            .value("2020")
            .build();

        PutJobTaggingRequest putJobTaggingRequest =
        PutJobTaggingRequest.builder()
            .jobId(jobId)
            .accountId(accountId)
            .tags(departmentTag, fiscalYearTag)
            .build();

        return asyncClient.putJobTagging(putJobTaggingRequest)
            .thenRun(() -> {
                System.out.println("Additional Tags were added to job " + jobId);
            })
            .exceptionally(ex -> {
                System.err.println("Failed to add tags to job: " +
ex.getMessage());
                throw new RuntimeException(ex); // Propagate the exception
            });
    }
```

- Per i dettagli sull'API, consulta la [PutJobTagging](#) sezione AWS SDK for Java 2.x API Reference.

Python

SDK per Python (Boto3)

Note

C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
def put_job_tags(self, job_id: str, account_id: str) -> None:
    """
```

Add tags to a batch job.

Args:

job_id (str): ID of the batch job

account_id (str): AWS account ID

"""

try:

```
self.s3control_client.put_job_tagging(
    AccountId=account_id,
    JobId=job_id,
    Tags=[
        {'Key': 'Environment', 'Value': 'Development'},
        {'Key': 'Team', 'Value': 'DataProcessing'}
    ]
)
```

```
print(f"Additional tags were added to job {job_id}")
```

except ClientError as e:

```
print(f"Error adding job tags: {e}")
```

```
raise
```

- Per i dettagli sull'API, consulta [PutJobTagging AWSSDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

TRY.

```
lo_s3c->putjobtagging(
    iv_accountid = iv_account_id
    iv_jobid      = iv_job_id
    it_tags       = VALUE /aws1/cl_s3cs3tag=>tt_s3tagset(
        ( NEW /aws1/cl_s3cs3tag(
            iv_key   = 'Environment'
            iv_value = 'Development' ) )
        ( NEW /aws1/cl_s3cs3tag(
```

```

        iv_key    = 'Team'
        iv_value = 'DataProcessing' ) )
    )
).
MESSAGE |Tags added to job { iv_job_id }| TYPE 'I'.
CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_nf.
CATCH /aws1/cx_s3ctoomanytagsex INTO DATA(lo_ex_tags).
MESSAGE lo_ex_tags->get_text( ) TYPE 'I'.
RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_tags.
CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_cli.
CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_srv.
ENDTRY.

```

- Per i dettagli sulle API, [PutJobTagging](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **UpdateJobPriority** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare UpdateJobPriority.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Informazioni di base](#)

CLI

AWS CLI

Come aggiornare la priorità di un processo Operazioni in batch Amazon S3

L'esempio `update-job-priority` seguente aggiorna il processo specificato in base a una nuova priorità.

```
aws s3control update-job-priority \
  --account-id 123456789012 \
  --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \
  --priority 52
```

Output:

```
{
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386",
  "Priority": 52
}
```

- Per i dettagli sull'API, consulta [UpdateJobPriority AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 * Updates the priority of a job asynchronously.
 *
 * @param jobId      the ID of the job to update
 * @param accountId the ID of the account associated with the job
 * @return a {@link CompletableFuture} that represents the asynchronous
 * operation, which completes when the job priority has been updated or an error
 * has occurred
```

```
    */
    public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
        UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .priority(60)
            .build();

        CompletableFuture<Void> future = new CompletableFuture<>();
        getAsyncClient().updateJobPriority(priorityRequest)
            .thenAccept(response -> {
                System.out.println("The job priority was updated");
                future.complete(null); // Complete the CompletableFuture on
successful execution
            })
            .exceptionally(ex -> {
                System.err.println("Failed to update job priority: " +
ex.getMessage());
                future.completeExceptionally(ex); // Complete the
CompletableFuture exceptionally on error
                return null; // Return null to handle the exception
            });

        return future;
    }
}
```

- Per i dettagli sull'API, consulta la [UpdateJobPriority](#) sezione AWS SDK for Java 2.x API Reference.

Python

SDK per Python (Boto3)

Note

C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).


```
def update_job_priority(self, job_id: str, account_id: str) -> None:
    """
    Update the priority of a batch job and start it.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")

        if current_status in ['Ready', 'Suspended']:
            self.s3control_client.update_job_priority(
                AccountId=account_id,
                JobId=job_id,
                Priority=60
            )
            print("The job priority was updated")

            try:
                self.s3control_client.update_job_status(
                    AccountId=account_id,
                    JobId=job_id,
                    RequestedJobStatus='Ready'
                )
                print("Job activated successfully")
            except ClientError as activation_error:
                print(f"Note: Could not activate job automatically: {activation_error}")
                print("Job priority was updated successfully. Job may need manual activation in the console.")
            elif current_status in ['Active', 'Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - priority cannot be updated")

            if current_status == 'Completing':
                print("Job is finishing up and will complete soon.")
            elif current_status == 'Complete':
                print("Job has already completed successfully.")
```

```

        else:
            print("Job is currently running.")
    else:
        print(f"Job is in '{current_status}' state - priority update not
allowed")

except ClientError as e:
    print(f"Error updating job priority: {e}")
    print("Continuing with the scenario...")
    return

```

- Per i dettagli sull'API, consulta [UpdateJobPriority AWS SDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

TRY.
    oo_result = lo_s3c->updatejobpriority( " oo_result is returned for
testing purposes.
    iv_accountid = iv_account_id
    iv_jobid      = iv_job_id
    iv_priority   = 60
    ).
    MESSAGE |Job { oo_result->get_jobid( ) } priority updated to { oo_result-
>get_priority( ) }| TYPE 'I'.
    CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_nf.
    CATCH /aws1/cx_s3cbadrequestex INTO DATA(lo_ex_bad).
    MESSAGE lo_ex_bad->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic

```

```

EXPORTING previous = lo_ex_bad.
CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
RAISE EXCEPTION TYPE /aws1/cx_rt_generic
EXPORTING previous = lo_ex_cli.
CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
RAISE EXCEPTION TYPE /aws1/cx_rt_generic
EXPORTING previous = lo_ex_srv.
ENDTRY.

```

- Per i dettagli sulle API, [UpdateJobPriority](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **UpdateJobStatus** con un AWS SDK o CLI

Gli esempi di codice seguenti mostrano come utilizzare UpdateJobStatus.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Informazioni di base](#)

CLI

AWS CLI

Come aggiornare lo stato di un processo Operazioni in batch Amazon S3

L'esempio update-job-status seguente annulla il processo specificato che è in attesa di approvazione.

```

aws s3control update-job-status \
  --account-id 123456789012 \
  --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \
  --requested-job-status Cancelled

```

Output:

```
{
  "Status": "Cancelled",
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386"
}
```

L'esempio `update-job-status` seguente conferma ed esegue il processo specificato che è in attesa di approvazione.

```
aws s3control update-job-status \
  --account-id 123456789012 \
  --job-id 5782949f-3301-4fb3-be34-8d5bab54dbca \
  --requested-job-status Ready
```

Output::

```
{
  "Status": "Ready",
  "JobId": "5782949f-3301-4fb3-
be34-8d5bab54dbca"
}
```

L'esempio `update-job-status` seguente annulla il processo in esecuzione specificato.

```
aws s3control update-job-status \
  --account-id 123456789012 \
  --job-id 5782949f-3301-4fb3-be34-8d5bab54dbca \
  --requested-job-status Cancelled
```

Output::

```
{
  "Status": "Cancelling",
  "JobId": "5782949f-3301-4fb3-be34-8d5bab54dbca"
}
```

- Per i dettagli sull'API, consulta [UpdateJobStatus AWS CLI Command Reference](#).

Java

SDK per Java 2.x

Note

C'è altro su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
/**
 * Cancels a job asynchronously.
 *
 * @param jobId The ID of the job to be canceled.
 * @param accountId The ID of the account associated with the job.
 * @return A {@link CompletableFuture} that completes when the job status has
 *         been updated to "CANCELLED".
 *         If an error occurs during the update, the returned future will
 *         complete exceptionally.
 */
public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId)
{
    UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
        .accountId(accountId)
        .jobId(jobId)
        .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
        .build();

    return asyncClient.updateJobStatus(updateJobStatusRequest)
        .thenAccept(updateJobStatusResponse -> {
            System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
        })
        .exceptionally(ex -> {
            System.err.println("Failed to cancel job: " + ex.getMessage());
            throw new RuntimeException(ex); // Propagate the exception
        });
}
```

- Per i dettagli sull'API, consulta la [UpdateJobStatus](#) sezione AWS SDK for Java 2.x API Reference.

Python

SDK per Python (Boto3)

Note

C'è di più su GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
def cancel_job(self, job_id: str, account_id: str) -> None:
    """
    Cancel an S3 batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")

        if current_status in ['Ready', 'Suspended', 'Active']:
            self.s3control_client.update_job_status(
                AccountId=account_id,
                JobId=job_id,
                RequestedJobStatus='Cancelled'
            )
            print(f"Job {job_id} was successfully canceled.")
        elif current_status in ['Completing', 'Complete']:
            print(f"Job is in '{current_status}' state - cannot be
cancelled")

        if current_status == 'Completing':
            print("Job is finishing up and will complete soon.")
```

```

        elif current_status == 'Complete':
            print("Job has already completed successfully.")
        else:
            print(f"Job is in '{current_status}' state - cancel not allowed")
    except ClientError as e:
        print(f"Error canceling job: {e}")
        raise

```

- Per i dettagli sull'API, consulta [UpdateJobStatus AWSSDK for Python \(Boto3\) API Reference](#).

SAP ABAP

SDK per SAP ABAP

Note

C'è altro da fare. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```

TRY.
    " iv_requested_status = 'Cancelled'
    oo_result = lo_s3c->updatejobstatus(      " oo_result is returned for
testing purposes.
        iv_accountid      = iv_account_id
        iv_jobid          = iv_job_id
        iv_requestedjobstatus = iv_requested_status
    ).
    MESSAGE |Job { oo_result->get_jobid( ) } status updated to { oo_result-
>get_status( ) }| TYPE 'I'.
    CATCH /aws1/cx_s3cjobstatusexception INTO DATA(lo_ex_js).
    MESSAGE lo_ex_js->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_js.
    CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_nf.
    CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).

```

```
MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.  
RAISE EXCEPTION TYPE /aws1/cx_rt_generic  
EXPORTING previous = lo_ex_cli.  
CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).  
MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.  
RAISE EXCEPTION TYPE /aws1/cx_rt_generic  
EXPORTING previous = lo_ex_srv.  
ENDTRY.
```

- Per i dettagli sulle API, [UpdateJobStatus](#) consulta AWS SDK for SAP ABAP API reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Esempi di codice per l'utilizzo di S3 Directory Bucket AWS SDK

I seguenti esempi di codice mostrano come utilizzare S3 Directory Buckets con un kit di sviluppo AWS software (SDK).

Nozioni di base: esempi di codice che mostrano come eseguire le operazioni essenziali all'interno di un servizio.

Le azioni sono estratti di codice da programmi più grandi e devono essere eseguite nel contesto. Sebbene le azioni mostrino come richiamare le singole funzioni del servizio, è possibile visualizzarle contestualizzate negli scenari correlati.

Scenari: esempi di codice che mostrano come eseguire un'attività specifica chiamando più funzioni all'interno dello stesso servizio o combinate con altri Servizi AWS.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Bucket di directory S3

- [Esempi di base per l'utilizzo di S3 Directory Bucket AWS SDK](#)
 - [Hello Amazon S3 directory buckets](#)
 - [Impara le basi di S3 Directory Buckets con un AWS SDK](#)

- [Azioni per l'utilizzo di S3 Directory Bucket AWS SDK](#)
 - [Utilizzare AbortMultipartUploadcon un AWS SDK](#)
 - [Utilizzare CompleteMultipartUploadcon un AWS SDK](#)
 - [Utilizzare CopyObjectcon un AWS SDK](#)
 - [Utilizzare CreateBucketcon un AWS SDK](#)
 - [Utilizzare CreateMultipartUploadcon un AWS SDK](#)
 - [Utilizzare CreateSessioncon un AWS SDK](#)
 - [Utilizzare DeleteBucketcon un AWS SDK](#)
 - [Utilizzare DeleteBucketEncryptioncon un AWS SDK](#)
 - [Utilizzare DeleteBucketPolicycon un AWS SDK](#)
 - [Utilizzare DeleteObjectcon un AWS SDK](#)
 - [Utilizzare DeleteObjectscon un AWS SDK](#)
 - [Utilizzare GetBucketEncryptioncon un AWS SDK](#)
 - [Utilizzare GetBucketPolicycon un AWS SDK](#)
 - [Utilizzare GetObjectcon un AWS SDK](#)
 - [Utilizzare GetObjectAttributescon un AWS SDK](#)
 - [Utilizzare HeadBucketcon un AWS SDK](#)
 - [Utilizzare HeadObjectcon un AWS SDK](#)
 - [Utilizzare ListDirectoryBucketscon un AWS SDK](#)
 - [Utilizzare ListMultipartUploadscon un AWS SDK](#)
 - [Usa ListObjectsV2 con un AWS SDK](#)
 - [Utilizzare ListPartscon un AWS SDK](#)
 - [Utilizzare PutBucketEncryptioncon un AWS SDK](#)
 - [Utilizzare PutBucketPolicycon un AWS SDK](#)
 - [Utilizzare PutObjectcon un AWS SDK](#)
 - [Utilizzare UploadPartcon un AWS SDK](#)
 - [Utilizzare UploadPartCopycon un AWS SDK](#)
- [Scenari per l'utilizzo di S3 Directory Bucket AWS SDK](#)
 - [Crea un URL predefinito per i bucket di directory Amazon S3 per ottenere un oggetto utilizzando un AWS SDK](#)

Esempi di base per l'utilizzo di S3 Directory Bucket AWS SDK

Gli esempi di codice seguenti mostrano come utilizzare le nozioni di base dei bucket di directory Amazon S3 con gli SDK AWS .

Esempi

- [Hello Amazon S3 directory buckets](#)
- [Impara le basi di S3 Directory Buckets con un AWS SDK](#)
- [Azioni per l'utilizzo di S3 Directory Bucket AWS SDK](#)
 - [Utilizzare AbortMultipartUploadcon un AWS SDK](#)
 - [Utilizzare CompleteMultipartUploadcon un AWS SDK](#)
 - [Utilizzare CopyObjectcon un AWS SDK](#)
 - [Utilizzare CreateBucketcon un AWS SDK](#)
 - [Utilizzare CreateMultipartUploadcon un AWS SDK](#)
 - [Utilizzare CreateSessioncon un AWS SDK](#)
 - [Utilizzare DeleteBucketcon un AWS SDK](#)
 - [Utilizzare DeleteBucketEncryptioncon un AWS SDK](#)
 - [Utilizzare DeleteBucketPolicycon un AWS SDK](#)
 - [Utilizzare DeleteObjectcon un AWS SDK](#)
 - [Utilizzare DeleteObjectscon un AWS SDK](#)
 - [Utilizzare GetBucketEncryptioncon un AWS SDK](#)
 - [Utilizzare GetBucketPolicycon un AWS SDK](#)
 - [Utilizzare GetObjectcon un AWS SDK](#)
 - [Utilizzare GetObjectAttributescon un AWS SDK](#)
 - [Utilizzare HeadBucketcon un AWS SDK](#)
 - [Utilizzare HeadObjectcon un AWS SDK](#)
 - [Utilizzare ListDirectoryBucketscon un AWS SDK](#)
 - [Utilizzare ListMultipartUploadscon un AWS SDK](#)
 - [Usa ListObjectsV2 con un AWS SDK](#)
 - [Utilizzare ListPartscon un AWS SDK](#)
 - [Utilizzare PutBucketEncryptioncon un AWS SDK](#)
 - [Utilizzare PutBucketPolicycon un AWS SDK](#)

- [Utilizzare PutObject con un AWS SDK](#)
- [Utilizzare UploadPart con un AWS SDK](#)
- [Utilizzare UploadPartCopy con un AWS SDK](#)

Hello Amazon S3 directory buckets

L'esempio di codice seguente mostra come iniziare a utilizzare i bucket di directory di Amazon S3.

Java

SDK per Java 2.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
package com.example.s3.directorybucket;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

/**
 * Before running this example:
 * <p>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have
 * not configured
 * authentication for SDKs and tools, see
 * https://docs.aws.amazon.com/sdkref/latest/guide/access.html in the AWS SDKs
 * and Tools Reference Guide.
 * <p>
 * You must have a runtime environment configured with the Java SDK.
 * See
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in
 * the Developer Guide if this is not set up.
 * <p>
 * To use S3 directory buckets, configure a gateway VPC endpoint. This is the
 * recommended method to enable directory bucket traffic without
 * requiring an internet gateway or NAT device. For more information on
 * configuring VPC gateway endpoints, visit
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-networking.html#s3-express-networking-vpc-gateway.
 * <p>
 * Directory buckets are available in specific AWS Regions and Zones. For
 * details on Regions and Zones supporting directory buckets, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-networking.html#s3-express-endpoints.
 */

public class HelloS3DirectoryBuckets {
    private static final Logger logger =
        LoggerFactory.getLogger(HelloS3DirectoryBuckets.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.

        Region region = Region.US_WEST_2;
        String zone = "usw2-az1";
        S3Client s3Client = createS3Client(region);

        try {
```

```

        // Create the directory bucket
        createDirectoryBucket(s3Client, bucketName, zone);
        logger.info("Created bucket: {}", bucketName);

        // List all directory buckets
        List<String> bucketNames = listDirectoryBuckets(s3Client);
        bucketNames.forEach(name -> logger.info("Bucket Name: {}", name));
    } catch (S3Exception e) {
        logger.error("An error occurred during S3 operations: {} - Error
code: {}",
                    e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
    } finally {
        try {
            // Delete the created bucket
            deleteDirectoryBucket(s3Client, bucketName);
            logger.info("Deleted bucket: {}", bucketName);
        } catch (S3Exception e) {
            logger.error("Failed to delete the bucket due to S3 error: {} -
Error code: {}",
                        e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
        } catch (RuntimeException e) {
            logger.error("Failed to delete the bucket due to unexpected
error: {}", e.getMessage(), e);
        } finally {
            s3Client.close();
        }
    }
}

/**
 * Creates a new S3 directory bucket in a specified Zone (For example, a
 * specified Availability Zone in this code example).
 *
 * @param s3Client    The S3 client used to create the bucket
 * @param bucketName The name of the bucket to be created
 * @param zone        The region where the bucket will be created
 * @throws S3Exception if there's an error creating the bucket
 */
public static void createDirectoryBucket(S3Client s3Client, String
bucketName, String zone) throws S3Exception {
    logger.info("Creating bucket: {}", bucketName);

```

```

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
            .location(LocationInfo.builder()
                .type(LocationType.AVAILABILITY_ZONE)
                .name(zone).build())
            .bucket(BucketInfo.builder()
                .type(BucketType.DIRECTORY)
                .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                .build())
            .build();
    try {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .createBucketConfiguration(bucketConfiguration).build();
        CreateBucketResponse response = s3Client.createBucket(bucketRequest);
        logger.info("Bucket created successfully with location: {}",
response.location());
    } catch (S3Exception e) {
        logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}

/**
 * Lists all S3 directory buckets.
 *
 * @param s3Client The S3 client used to interact with S3
 * @return A list of bucket names
 */
public static List<String> listDirectoryBuckets(S3Client s3Client) {
    logger.info("Listing all directory buckets");

    try {
        // Create a ListBucketsRequest
        ListDirectoryBucketsRequest listBucketsRequest =
ListDirectoryBucketsRequest.builder().build();

        // Retrieve the list of buckets
        ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listBucketsRequest);

        // Extract bucket names

```

```

        List<String> bucketNames = response.buckets().stream()
            .map(Bucket::name)
            .collect(Collectors.toList());

        return bucketNames;
    } catch (S3Exception e) {
        logger.error("Failed to list buckets: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}

/**
 * Deletes the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the bucket to delete
 */
public static void deleteDirectoryBucket(S3Client s3Client, String
bucketName) {
    try {
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
            .bucket(bucketName)
            .build();
        s3Client.deleteBucket(deleteBucketRequest);
    } catch (S3Exception e) {
        logger.error("Failed to delete bucket: " + bucketName + " - Error
code: " + e.awsErrorDetails().errorCode(),
            e);
        throw e;
    }
}
}
}

```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK for Java 2.x .
 - [CreateBucket](#)
 - [ListDirectoryBuckets](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Impara le basi di S3 Directory Buckets con un AWS SDK

Gli esempi di codice seguenti mostrano come:

- Configurare un VPC e un endpoint VPC.
- Configurare policy, ruoli e utente da utilizzare con i bucket di directory S3 e la classe di archiviazione S3 Express One Zone.
- Creare due client S3.
- Creare due bucket.
- Creare un oggetto e copiarlo.
- Descrivere la differenza di prestazioni.
- Popolare i bucket per mostrare la differenza lessicografica.
- Richiedere all'utente di verificare se eseguire o meno la pulizia delle risorse.

Java

SDK per Java 2.x

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegue uno scenario interattivo che dimostra le funzionalità di Amazon S3.

```
public class S3DirectoriesScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
    "-");

    private static final Logger logger =
    LoggerFactory.getLogger(S3DirectoriesScenario.class);
    static Scanner scanner = new Scanner(System.in);
```



```
private static S3AsyncClient mS3RegularClient;
private static S3AsyncClient mS3ExpressClient;

private static String mdirectoryBucketName;
private static String mregularBucketName;

private static String stackName = "cfn-stack-s3-express-basics--" +
UUID.randomUUID();

private static String regularUser = "";
private static String vpcId = "";
private static String expressUser = "";

private static String vpcEndpointId = "";

private static final S3DirectoriesActions s3DirectoriesActions = new
S3DirectoriesActions();

public static void main(String[] args) {
    try {
        s3ExpressScenario();
    } catch (RuntimeException e) {
        logger.info(e.getMessage());
    }
}

// Runs the scenario.
private static void s3ExpressScenario() {
    logger.info(DASHES);
    logger.info("Welcome to the Amazon S3 Express Basics demo using AWS SDK
for Java V2.");
    logger.info(""""
        Let's get started! First, please note that S3 Express One Zone works
best when working within the AWS infrastructure,
        specifically when working in the same Availability Zone (AZ). To see
the best results in this example and when you implement
        directory buckets into your infrastructure, it is best to put your
compute resources in the same AZ as your directory
        bucket.
        """);
    waitForInputToContinue(scanner);
    logger.info(DASHES);
}
```

```
// Create an optional VPC and create 2 IAM users.
UserNames userNames = createVpcUsers();
String expressUserName = userNames.getExpressUserName();
String regularUserName = userNames.getRegularUserName();

// Set up two S3 clients, one regular and one express,
// and two buckets, one regular and one directory.
setUpClientsAndBuckets(expressUserName, regularUserName);

// Create an S3 session for the express S3 client and add objects to the
buckets.
logger.info("Now let's add some objects to our buckets and demonstrate
how to work with S3 Sessions.");
waitForInputToContinue(scanner);
String bucketObject = createSessionAdd0bjects();

// Demonstrate performance differences between regular and directory
buckets.
demonstratePerformance(bucketObject);

// Populate the buckets to show the lexicographical difference between
// regular and express buckets.
showLexicographicalDifferences(bucketObject);

logger.info(DASHES);
logger.info("That's it for our tour of the basic operations for S3
Express One Zone.");
logger.info("Would you like to cleanUp the AWS resources? (y/n): ");
String response = scanner.next().trim().toLowerCase();
if (response.equals("y")) {
    cleanUp(stackName);
}
}

/*
Delete resources created by this scenario.
*/
public static void cleanUp(String stackName) {
    try {
        if (mdirectoryBucketName != null) {

s3DirectoriesActions.deleteBucketAnd0bjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
        }
    }
}
```

```

        logger.info("Deleted directory bucket " + mdirectoryBucketName);
        mdirectoryBucketName = null;
        if (mregularBucketName != null) {

s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3RegularClient,
mregularBucketName).join();
        }
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
        }

        logger.info("Deleted regular bucket " + mregularBucketName);
        mregularBucketName = null;
        CloudFormationHelper.destroyCloudFormationStack(stackName);
    }

    private static void showLexicographicalDifferences(String bucketObject) {
        logger.info(DASHES);
        logger.info(""""
            7. Populate the buckets to show the lexicographical (alphabetical)
difference
store
name
            "Directory". Where regular buckets store their key/value pairs in a
flat manner, directory buckets use actual directories/folders.
This allows for more rapid indexing, traversing, and therefore
retrieval times!

            The more segmented your bucket is, with lots of
directories, sub-directories, and objects, the more efficient it
becomes.

            This structural difference also causes `ListObject` operations to
behave
more
            differently, which can cause unexpected results. Let's add a few
objects in sub-directories to see how the output of

```

```
        ListObjects changes.  
        "");  
  
    waitForInputToContinue(scanner);  
  
    // Populate a few more files in each bucket so that we can use  
    // ListObjects and show the difference.  
    String otherObject = "other/" + bucketObject;  
    String altObject = "alt/" + bucketObject;  
    String otherAltObject = "other/alt/" + bucketObject;  
  
    try {  
        s3DirectoriesActions.putObjectAsync(mS3RegularClient,  
mregularBucketName, otherObject, "").join();  
        s3DirectoriesActions.putObjectAsync(mS3ExpressClient,  
mdirectoryBucketName, otherObject, "").join();  
        s3DirectoriesActions.putObjectAsync(mS3RegularClient,  
mregularBucketName, altObject, "").join();  
        s3DirectoriesActions.putObjectAsync(mS3ExpressClient,  
mdirectoryBucketName, altObject, "").join();  
        s3DirectoriesActions.putObjectAsync(mS3RegularClient,  
mregularBucketName, otherAltObject, "").join();  
        s3DirectoriesActions.putObjectAsync(mS3ExpressClient,  
mdirectoryBucketName, otherAltObject, "").join();  
  
        } catch (CompletionException ce) {  
            Throwable cause = ce.getCause();  
            if (cause instanceof NoSuchBucketException) {  
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);  
            } else {  
                logger.error("An unexpected error occurred: {}",  
cause.getMessage(), ce);  
            }  
            return;  
        }  
  
        try {  
            // List objects in both S3 buckets.  
            List<String> dirBucketObjects =  
s3DirectoriesActions.listObjectsAsync(mS3ExpressClient,  
mdirectoryBucketName).join();  
            List<String> regBucketObjects =  
s3DirectoriesActions.listObjectsAsync(mS3RegularClient,  
mregularBucketName).join();
```

```

        logger.info("Directory bucket content");
        for (String obj : dirBucketObjects) {
            logger.info(obj);
        }

        logger.info("Regular bucket content");
        for (String obj : regBucketObjects) {
            logger.info(obj);
        }
    } catch (CompletionException e) {
        logger.error("Async operation failed: {} ",
e.getCause().getMessage());
        return;
    }

    logger.info("""
        Notice how the regular bucket lists objects in lexicographical order,
        while the directory bucket does not. This is
        because the regular bucket considers the whole "key" to be the object
        identifier, while the directory bucket actually
        creates directories and uses the object "key" as a path to the
        object.
        """);
    waitForInputToContinue(scanner);
}

/**
 * Demonstrates the performance difference between downloading an object from
 * a directory bucket and a regular bucket.
 *
 * <p>This method:
 * <ul>
 *     <li>Prompts the user to choose the number of downloads (default is
1,000).</li>
 *     <li>Downloads the specified object from the directory bucket and
measures the total time.</li>
 *     <li>Downloads the same object from the regular bucket and measures the
total time.</li>
 *     <li>Compares the time differences and prints the results.</li>
 * </ul>
 *
 * <p>Note: The performance difference will be more pronounced if this
example is run on an EC2 instance

```

```

    * in the same Availability Zone as the buckets.
    *
    * @param bucketObject the name of the object to download
    */
private static void demonstratePerformance(String bucketObject) {
    logger.info(DASHES);
    logger.info("6. Demonstrate the performance difference.");
    logger.info("""
        Now, let's do a performance test. We'll download the same object from
each
        bucket repeatedly and compare the total time needed.

        Note: the performance difference will be much more pronounced if this
        example is run in an EC2 instance in the same Availability Zone as
        the bucket.
        """);
    waitForInputToContinue(scanner);

    int downloads = 1000; // Default value.
    logger.info("The default number of downloads of the same object for this
example is set at " + downloads + ".");

    // Ask if the user wants to download a different number.
    logger.info("Would you like to download the file a different number of
times? (y/n): ");
    String response = scanner.next().trim().toLowerCase();
    if (response.equals("y")) {
        int maxDownloads = 1_000_000;

        // Ask for a valid number of downloads.
        while (true) {
            logger.info("Enter a number between 1 and " + maxDownloads + "
for the number of downloads: ");
            if (scanner.hasNextInt()) {
                downloads = scanner.nextInt();
                if (downloads >= 1 && downloads <= maxDownloads) {
                    break;
                } else {
                    logger.info("Please enter a number between 1 and " +
maxDownloads + ".");
                }
            } else {
                logger.info("Invalid input. Please enter a valid integer.");
                scanner.next();
            }
        }
    }
}

```

```
        }
    }

    logger.info("You have chosen to download {} items.", downloads);
} else {
    logger.info("No changes made. Using default downloads: {}",
downloads);
}
// Simulating the download process for the directory bucket.
logger.info("Downloading from the directory bucket.");
long directoryTimeStart = System.nanoTime();
for (int index = 0; index < downloads; index++) {
    if (index % 50 == 0) {
        logger.info("Download " + index + " of " + downloads);
    }

    try {
        // Get the object from the directory bucket.
        s3DirectoriesActions.getObjectAsync(mS3ExpressClient,
mdirectoryBucketName, bucketObject).join();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchKeyException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }
}
long directoryTimeDifference = System.nanoTime() - directoryTimeStart;

// Download from the regular bucket.
logger.info("Downloading from the regular bucket.");
long normalTimeStart = System.nanoTime();
for (int index = 0; index < downloads; index++) {
    if (index % 50 == 0) {
        logger.info("Download " + index + " of " + downloads);
    }

    try {
```

```

        s3DirectoriesActions.getObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject).join();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchKeyException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }
}

    long normalTimeDifference = System.nanoTime() - normalTimeStart;
    logger.info("The directory bucket took " + directoryTimeDifference
+ " nanoseconds, while the regular bucket took " + normalTimeDifference + "
nanoseconds.");
    long difference = normalTimeDifference - directoryTimeDifference;
    logger.info("That's a difference of " + difference + " nanoseconds, or");
    logger.info(difference / 1_000_000_000.0 + " seconds.");

    if (difference < 0) {
        logger.info("The directory buckets were slower. This can happen if
you are not running on the cloud within a VPC.");
    }
    waitForInputToContinue(scanner);
}

private static String createSessionAddObjects() {
    logger.info(DASHES);
    logger.info("""
        5. Create an object and copy it.
        We'll create an object consisting of some text and upload it to the
        regular bucket.
        """);
    waitForInputToContinue(scanner);

    String bucketObject = "basic-text-object.txt";
    try {
        s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject, "Look Ma, I'm a bucket!").join();

```



```

        s3DirectoriesActions.createSessionAsync(mS3ExpressClient,
mdirectoryBucketName).join();

        // Copy the object to the destination S3 bucket.
        s3DirectoriesActions.copyObjectAsync(mS3ExpressClient,
mregularBucketName, bucketObject, mdirectoryBucketName, bucketObject).join();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof S3Exception) {
            logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
    }
    logger.info("""
        It worked! This is because the S3Client that performed the copy
operation
        is the expressClient using the credentials for the user with
permission to
        work with directory buckets.

        It's important to remember the user permissions when interacting
with
        directory buckets. Instead of validating permissions on every call
as
        regular buckets do, directory buckets utilize the user credentials
and session
        token to validate. This allows for much faster connection speeds on
every call.
        For single calls, this is low, but for many concurrent calls
        this adds up to a lot of time saved.
        """);
    waitForInputToContinue(scanner);
    return bucketObject;
}

/**
 * Creates VPC users for the S3 Express One Zone scenario.
 * <p>
 * This method performs the following steps:
 * <ol>

```

```

    *      <li>Optionally creates a new VPC and VPC Endpoint if the application
is running in an EC2 instance in the same Availability Zone as the directory
buckets.</li>
    *      <li>Creates two IAM users: one with S3 Express One Zone permissions
and one without.</li>
    * </ol>
    *
    * @return a {@link UserNames} object containing the names of the created IAM
users
    */
    public static UserNames createVpcUsers() {
        /*
        Optionally create a VPC.
        Create two IAM users, one with S3 Express One Zone permissions and one
without.
        */
        logger.info(DASHES);
        logger.info("""
            1. First, we'll set up a new VPC and VPC Endpoint if this program is
running in an EC2 instance in the same AZ as your\s
            directory buckets will be. Are you running this in an EC2 instance
located in the same AZ as your intended directory buckets?
            """);

        logger.info("Do you want to setup a VPC Endpoint? (y/n)");
        String endpointAns = scanner.nextLine().trim();
        if (endpointAns.equalsIgnoreCase("y")) {
            logger.info("""
                Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to.
                """);
            try {
                s3DirectoriesActions.setupVPCAsync().join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof Ec2Exception) {
                    logger.error("IamException occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
            }
            waitForInputToContinue(scanner);

```

```

    } else {
        logger.info("Skipping the VPC setup. Don't forget to use this in
production!");
    }
    logger.info(DASHES);
    logger.info("""
        2. Create a RegularUser and ExpressUser by using the AWS CDK.
        One IAM User, named RegularUser, will have permissions to work only
        with regular buckets and one IAM user, named ExpressUser, will have
        permissions to work only with directory buckets.
        """);
    waitForInputToContinue(scanner);

    // Create two users required for this scenario.
    Map<String, String> stackOutputs = createUsersUsingCDK(stackName);
    regularUser = stackOutputs.get("RegularUser");
    expressUser = stackOutputs.get("ExpressUser");

    UserNames names = new UserNames();
    names.setRegularUserName(regularUser);
    names.setExpressUserName(expressUser);
    return names;
}

/**
 * Creates users using AWS CloudFormation.
 *
 * @return a {@link Map} of String keys and String values representing the
stack outputs,
 * which may include user-related information such as user names and IDs.
 */
public static Map<String, String> createUsersUsingCDK(String stackName) {
    logger.info("We'll use an AWS CloudFormation template to create the IAM
users and policies.");
    CloudFormationHelper.deployCloudFormationStack(stackName);
    return CloudFormationHelper.getStackOutputsAsync(stackName).join();
}

/**
 * Sets up the necessary clients and buckets for the S3 Express service.
 *
 * @param expressUserName the username for the user with S3 Express
permissions

```

```

    * @param regularUserName the username for the user with regular S3
permissions
    */
    public static void setupClientsAndBuckets(String expressUserName, String
regularUserName) {
        Scanner locscanner = new Scanner(System.in);
        String accessKeyIdforRegUser;
        String secretAccessforRegUser;
        try {
            CreateAccessKeyResponse keyResponse =
s3DirectoriesActions.createAccessKeyAsync(regularUserName).join();
            accessKeyIdforRegUser = keyResponse.accessKey().accessKeyId();
            secretAccessforRegUser = keyResponse.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(),
ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }

        String accessKeyIdforExpressUser;
        String secretAccessforExpressUser;
        try {
            CreateAccessKeyResponse keyResponseExpress =
s3DirectoriesActions.createAccessKeyAsync(expressUserName).join();
            accessKeyIdforExpressUser =
keyResponseExpress.accessKey().accessKeyId();
            secretAccessforExpressUser =
keyResponseExpress.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(),
ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }
    }

```

```

    }

    logger.info(DASHES);
    logger.info("""
        3. Create two S3Clients; one uses the ExpressUser's credentials and
one uses the RegularUser's credentials.
        The 2 S3Clients will use different credentials.
        """);
    waitForInputToContinue(locscanner);
    try {
        mS3RegularClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforRegUser,
secretAccessforRegUser).join();
        mS3ExpressClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforExpressUser,
secretAccessforExpressUser).join();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof IllegalArgumentException) {
            logger.error("An invalid argument exception occurred: {}",
cause.getMessage(), ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }

    logger.info("""
        We can now use the ExpressUser client to make calls to S3 Express
operations.
        """);
    waitForInputToContinue(locscanner);
    logger.info(DASHES);
    logger.info("""
        4. Create two buckets.
        Now we will create a directory bucket which is the linchpin of the S3
Express One Zone service. Directory buckets
        behave differently from regular S3 buckets which we will explore
here. We'll also create a regular bucket, put
        an object into the regular bucket, and copy it to the directory
bucket.
        """);

```

```

        logger.info("""
            Now, let's choose an availability zone (AZ) for the directory
bucket.

            We'll choose one that is supported.
            """);
        String zoneId;
        String regularBucketName;
        try {
            zoneId = s3DirectoriesActions.selectAvailabilityZoneIdAsync().join();
            regularBucketName = "amzn-s3-demo-bucket"; // Replace with your
bucket name.
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof Ec2Exception) {
                logger.error("EC2Exception occurred: {}", cause.getMessage(),
ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }
        logger.info("""
            Now, let's create the actual directory bucket, as well as a regular
bucket."
            """);

        String directoryBucketName = "amzn-s3-demo-bucket--" + zoneId + "--x-
s3"; // Replace with your directory bucket name.
        try {
            s3DirectoriesActions.createDirectoryBucketAsync(mS3ExpressClient,
directoryBucketName, zoneId).join();
            logger.info("Created directory bucket {}", directoryBucketName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }
    }
}

```

```

        // Assign to the data member.
        mdirectoryBucketName = directoryBucketName;
        try {
            s3DirectoriesActions.createBucketAsync(mS3RegularClient,
regularBucketName).join();
            logger.info("Created regular bucket {} ", regularBucketName);
            mregularBucketName = regularBucketName;
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                return;
            }
        }
        logger.info("Great! Both buckets were created.");
        waitForInputToContinue(locscanner);
    }

    /**
     * Creates an asynchronous S3 client with the specified access key and secret
     * access key.
     *
     * @param accessKeyId      the AWS access key ID
     * @param secretAccessKey the AWS secret access key
     * @return a {@link CompletableFuture} that asynchronously creates the S3
     * client
     * @throws IllegalArgumentException if the access key ID or secret access key
     * is null
     */
    public static CompletableFuture<S3AsyncClient>
createS3ClientWithAccessKeyAsync(String accessKeyId, String secretAccessKey) {
        return CompletableFuture.supplyAsync(() -> {
            // Validate input parameters
            if (accessKeyId == null || accessKeyId.isBlank() || secretAccessKey
== null || secretAccessKey.isBlank()) {
                throw new IllegalArgumentException("Access Key ID and Secret
Access Key must not be null or empty");
            }
        })
    }

```

```

        AwsBasicCredentials awsCredentials =
        AwsBasicCredentials.create(accessKeyId, secretAccessKey);
        return S3AsyncClient.builder()

        .credentialsProvider(StaticCredentialsProvider.create(awsCredentials))
            .region(Region.US_WEST_2)
            .build();
    });
}

private static void waitForInputToContinue(Scanner scanner) {
    while (true) {
        logger.info("");
        logger.info("Enter 'c' followed by <ENTER> to continue:");
        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
            logger.info("Continuing with the program...");
            logger.info("");
            break;
        } else {
            logger.info("Invalid input. Please try again.");
        }
    }
}
}
}

```

Una classe wrapper per i metodi dell'SDK di Amazon S3.

```

public class S3DirectoriesActions {

    private static IamAsyncClient iamAsyncClient;

    private static Ec2AsyncClient ec2AsyncClient;
    private static final Logger logger =
    LoggerFactory.getLogger(S3DirectoriesActions.class);

    private static IamAsyncClient getIAMAsyncClient() {
        if (iamAsyncClient == null) {
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(100)
                .connectionTimeout(Duration.ofSeconds(60))

```



```
        .readTimeout(Duration.ofSeconds(60))
        .writeTimeout(Duration.ofSeconds(60))
        .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofMinutes(2))
        .apiCallAttemptTimeout(Duration.ofSeconds(90))
        .retryStrategy(RetryMode.STANDARD)
        .build();

        iamAsyncClient = IamAsyncClient.builder()
        .httpClient(httpClient)
        .overrideConfiguration(overrideConfig)
        .build();
    }
    return iamAsyncClient;
}

private static Ec2AsyncClient getEc2AsyncClient() {
    if (ec2AsyncClient == null) {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
        .maxConcurrency(100)
        .connectionTimeout(Duration.ofSeconds(60))
        .readTimeout(Duration.ofSeconds(60))
        .writeTimeout(Duration.ofSeconds(60))
        .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofMinutes(2))
        .apiCallAttemptTimeout(Duration.ofSeconds(90))
        .retryStrategy(RetryMode.STANDARD)
        .build();

        ec2AsyncClient = Ec2AsyncClient.builder()
        .httpClient(httpClient)
        .region(Region.US_WEST_2)
        .overrideConfiguration(overrideConfig)
        .build();
    }
    return ec2AsyncClient;
}
```

```

/**
 * Deletes the specified S3 bucket and all the objects within it
 * asynchronously.
 *
 * @param s3AsyncClient the S3 asynchronous client to use for the operations
 * @param bucketName the name of the S3 bucket to be deleted
 * @return a {@link CompletableFuture} that completes with a {@link
 * WaiterResponse} containing the
 *
 *         {@link HeadBucketResponse} when the bucket has been successfully
 * deleted
 *
 * @throws CompletionException if there was an error deleting the bucket or
 * its objects
 */
public CompletableFuture<WaiterResponse<HeadBucketResponse>>
deleteBucketAndObjectsAsync(S3AsyncClient s3AsyncClient, String bucketName) {
    ListObjectsV2Request listRequest = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .build();

    return s3AsyncClient.listObjectsV2(listRequest)
        .thenCompose(listResponse -> {
            if (!listResponse.contents().isEmpty()) {
                List<ObjectIdentifier> objectIdentifiers =
listResponse.contents().stream()
                    .map(s3Object ->
ObjectIdentifier.builder().key(s3Object.key()).build())
                    .collect(Collectors.toList());

                DeleteObjectsRequest deleteRequest =
DeleteObjectsRequest.builder()
                    .bucket(bucketName)

.delete(Delete.builder().objects(objectIdentifiers).build())
                    .build();

                return s3AsyncClient.deleteObjects(deleteRequest)
                    .thenAccept(deleteResponse -> {
                        if (!deleteResponse.errors().isEmpty()) {
                            deleteResponse.errors().forEach(error ->
                                logger.error("Couldn't delete object " +
error.key() + ". Reason: " + error.message()));
                        }
                    });
            }
        });
}

```

```

        return CompletableFuture.completedFuture(null);
    })
    .thenCompose(ignored -> {
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
        .bucket(bucketName)
        .build();
        return s3AsyncClient.deleteBucket(deleteBucketRequest);
    })
    .thenCompose(ignored -> {
        S3AsyncWaiter waiter = s3AsyncClient.waiter();
        HeadBucketRequest headBucketRequest =
HeadBucketRequest.builder().bucket(bucketName).build();
        return waiter.waitUntilBucketNotExists(headBucketRequest);
    })
    .whenComplete((ignored, exception) -> {
        if (exception != null) {
            Throwable cause = exception.getCause();
            if (cause instanceof S3Exception) {
                throw new CompletionException("Error deleting bucket: " +
bucketName, cause);
            }
            throw new CompletionException("Failed to delete bucket and
objects: " + bucketName, exception);
        }
        logger.info("Bucket deleted successfully: " + bucketName);
    });
}

/**
 * Lists the objects in an S3 bucket asynchronously.
 *
 * @param s3Client the S3 async client to use for the operation
 * @param bucketName the name of the S3 bucket containing the objects to list
 * @return a {@link CompletableFuture} that contains the list of object keys
in the specified bucket
 */
public CompletableFuture<List<String>> listObjectsAsync(S3AsyncClient
s3Client, String bucketName) {
    ListObjectsV2Request request = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .build();

    return s3Client.listObjectsV2(request)

```

```

        .thenApply(response -> response.contents().stream()
            .map(S3Object::key)
            .toList())
        .whenComplete((result, exception) -> {
            if (exception != null) {
                throw new CompletionException("Couldn't list objects in
bucket: " + bucketName, exception);
            }
        });
    }

/**
 * Retrieves an object from an Amazon S3 bucket asynchronously.
 *
 * @param s3Client    the S3 async client to use for the operation
 * @param bucketName the name of the S3 bucket containing the object
 * @param keyName     the unique identifier (key) of the object to retrieve
 * @return a {@link CompletableFuture} that, when completed, contains the
object's content as a {@link ResponseBytes} of {@link GetObjectResponse}
 */
    public CompletableFuture<ResponseBytes<GetObjectResponse>>
getObjectAsync(S3AsyncClient s3Client, String bucketName, String keyName) {
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        // Get the object asynchronously and transform it into a byte array
        return s3Client.getObject(objectRequest,
AsyncResponseTransformer.toBytes())
            .exceptionally(exception -> {
                Throwable cause = exception.getCause();
                if (cause instanceof NoSuchKeyException) {
                    throw new CompletionException("Failed to get the object.
Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                }
                throw new CompletionException("Failed to get the object",
exception);
            });
    }

/**
 * Asynchronously copies an object from one S3 bucket to another.
 *

```

```

    * @param s3Client          the S3 async client to use for the copy
operation
    * @param sourceBucket      the name of the source bucket
    * @param sourceKey         the key of the object to be copied in the source
bucket
    * @param destinationBucket the name of the destination bucket
    * @param destinationKey    the key of the copied object in the destination
bucket
    * @return a {@link CompletableFuture} that completes when the copy operation
is finished
    */
    public CompletableFuture<Void> copyObjectAsync(S3AsyncClient s3Client, String
sourceBucket, String sourceKey, String destinationBucket, String destinationKey)
{
    CopyObjectRequest copyRequest = CopyObjectRequest.builder()
        .sourceBucket(sourceBucket)
        .sourceKey(sourceKey)
        .destinationBucket(destinationBucket)
        .destinationKey(destinationKey)
        .build();

    return s3Client.copyObject(copyRequest)
        .thenRun(() -> logger.info("Copied object '" + sourceKey + "' from
bucket '" + sourceBucket + "' to bucket '" + destinationBucket + "'"))
        .whenComplete((ignored, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof S3Exception) {
                    throw new CompletionException("Couldn't
copy object '" + sourceKey + "' from bucket '" + sourceBucket + "'
to bucket '" + destinationBucket + "'. Reason: " + ((S3Exception)
cause).awsErrorDetails().errorMessage(), cause);
                }
                throw new CompletionException("Failed to copy object",
exception);
            }
        });
}

/**
 * Asynchronously creates a session for the specified S3 bucket.
 *
 * @param s3Client the S3 asynchronous client to use for creating the
session

```

```

    * @param bucketName the name of the S3 bucket for which to create the
    session
    * @return a {@link CompletableFuture} that completes when the session is
    created, or throws a {@link CompletionException} if an error occurs
    */
    public CompletableFuture<CreateSessionResponse>
    createSessionAsync(S3AsyncClient s3Client, String bucketName) {
        CreateSessionRequest request = CreateSessionRequest.builder()
            .bucket(bucketName)
            .build();

        return s3Client.createSession(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Couldn't create the
session. Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(),
cause);
                    }
                    throw new CompletionException("Unexpected error occurred
while creating session", exception);
                }
                logger.info("Created session for bucket: " + bucketName);
            });
    }

    /**
    * Creates a new S3 directory bucket in a specified Zone (For example, a
    * specified Availability Zone in this code example).
    *
    * @param s3Client The asynchronous S3 client used to create the bucket
    * @param bucketName The name of the bucket to be created
    * @param zone The Availability Zone where the bucket will be created
    * @throws CompletionException if there's an error creating the bucket
    */
    public CompletableFuture<CreateBucketResponse>
    createDirectoryBucketAsync(S3AsyncClient s3Client, String bucketName, String
zone) {
        logger.info("Creating bucket: " + bucketName);

        CreateBucketConfiguration bucketConfiguration =
        CreateBucketConfiguration.builder()

```

```

        .location(LocationInfo.builder()
            .type(LocationType.AVAILABILITY_ZONE)
            .name(zone)
            .build())
        .bucket(BucketInfo.builder()
            .type(BucketType.DIRECTORY)
            .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
            .build())
        .build();

CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
    .bucket(bucketName)
    .createBucketConfiguration(bucketConfiguration)
    .build();

return s3Client.createBucket(bucketRequest)
    .whenComplete((response, exception) -> {
        if (exception != null) {
            Throwable cause = exception.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                throw new CompletionException("The bucket already exists:
" + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
            }
            throw new CompletionException("Unexpected error occurred
while creating bucket", exception);
        }
        logger.info("Bucket created successfully with location: " +
response.location());
    });
}

/**
 * Creates an S3 bucket asynchronously.
 *
 * @param s3Client    the S3 async client to use for the bucket creation
 * @param bucketName  the name of the S3 bucket to create
 * @return a {@link CompletableFuture} that completes with the {@link
WaiterResponse} containing the {@link HeadBucketResponse}
 *         when the bucket is successfully created
 * @throws CompletionException if there's an error creating the bucket
 */
public CompletableFuture<WaiterResponse<HeadBucketResponse>>
createBucketAsync(S3AsyncClient s3Client, String bucketName) {
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()

```

```

        .bucket(bucketName)
        .build();

    return s3Client.createBucket(bucketRequest)
        .thenCompose(response -> {
            S3AsyncWaiter s3Waiter = s3Client.waiter();
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();
            return s3Waiter.waitUntilBucketExists(bucketRequestWait);
        })
        .whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof BucketAlreadyExistsException) {
                    throw new CompletionException("The S3 bucket exists: " +
cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);
                }
            }
            logger.info(bucketName + " is ready");
        });
    }

/**
 * Uploads an object to an Amazon S3 bucket asynchronously.
 *
 * @param s3Client    the S3 async client to use for the upload
 * @param bucketName  the destination S3 bucket name
 * @param bucketObject the name of the object to be uploaded
 * @param text        the content to be uploaded as the object
 */
    public CompletableFuture<PutObjectResponse> putObjectAsync(S3AsyncClient
s3Client, String bucketName, String bucketObject, String text) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(bucketObject)
            .build();

        return s3Client.putObject(objectRequest,
AsyncRequestBody.fromString(text))
            .whenComplete((response, exception) -> {

```



```

        if (exception != null) {
            Throwable cause = exception.getCause();
            if (cause instanceof NoSuchBucketException) {
                throw new CompletionException("The S3 bucket does not
exist: " + cause.getMessage(), cause);
            } else {
                throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);
            }
        }
    });
}

/**
 * Creates an AWS IAM access key asynchronously for the specified user name.
 *
 * @param userName the name of the IAM user for whom to create the access key
 * @return a {@link CompletableFuture} that completes with the {@link
CreateAccessKeyResponse} containing the created access key
 */
public CompletableFuture<CreateAccessKeyResponse> createAccessKeyAsync(String
userName) {
    CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
        .userName(userName)
        .build();

    return getIAMAsyncClient().createAccessKey(request)
        .whenComplete((response, exception) -> {
            if (response != null) {
                logger.info("Access Key Created.");
            } else {
                if (exception == null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof IamException) {
                        throw new CompletionException("IAM error while
creating access key: " + cause.getMessage(), cause);
                    } else {
                        throw new CompletionException("Failed to create
access key: " + exception.getMessage(), exception);
                    }
                }
            }
        });
}
}

```

```

/**
 * Asynchronously selects an Availability Zone ID from the available EC2
zones.
 *
 * @return A {@link CompletableFuture} that resolves to the selected
Availability Zone ID.
 * @throws CompletionException if an error occurs during the request or
processing.
 */
public CompletableFuture<String> selectAvailabilityZoneIdAsync() {
    DescribeAvailabilityZonesRequest zonesRequest =
DescribeAvailabilityZonesRequest.builder()
        .build();

    return getEc2AsyncClient().describeAvailabilityZones(zonesRequest)
        .thenCompose(response -> {
            List<AvailabilityZone> zonesList = response.availabilityZones();
            if (zonesList.isEmpty()) {
                logger.info("No availability zones found.");
                return CompletableFuture.completedFuture(null); // Return
null if no zones are found
            }

            List<String> zoneIds = zonesList.stream()
                .map(AvailabilityZone::zoneId) // Get the zoneId (e.g.,
"usw2-az1")
                .toList();

            return CompletableFuture.supplyAsync(() ->
promptUserForZoneSelection(zonesList, zoneIds))
                .thenApply(selectedZone -> {
                    // Return only the selected Zone ID (e.g., "usw2-az1").
                    return selectedZone.zoneId();
                });
        })
        .whenComplete((result, exception) -> {
            if (exception == null) {
                if (result != null) {
                    logger.info("Selected Availability Zone ID: " + result);
                } else {
                    logger.info("No availability zone selected.");
                }
            } else {
                // else {

```

```

        Throwable cause = exception.getCause();
        if (cause instanceof Ec2Exception) {
            throw new CompletionException("EC2 error while selecting
availability zone: " + cause.getMessage(), cause);
        }
        throw new CompletionException("Failed to select availability
zone: " + exception.getMessage(), exception);
    }
});
}

/**
 * Prompts the user to select an Availability Zone from the given list.
 *
 * @param zonesList the list of Availability Zones
 * @param zoneIds the list of zone IDs
 * @return the selected Availability Zone
 */
private static AvailabilityZone
promptUserForZoneSelection(List<AvailabilityZone> zonesList, List<String>
zoneIds) {
    Scanner scanner = new Scanner(System.in);
    int index = -1;

    while (index < 0 || index >= zoneIds.size()) {
        logger.info("Select an availability zone:");
        IntStream.range(0, zoneIds.size()).forEach(i ->
            logger.info(i + ": " + zoneIds.get(i))
        );

        logger.info("Enter the number corresponding to your choice: ");
        if (scanner.hasNextInt()) {
            index = scanner.nextInt();
        } else {
            scanner.next();
        }
    }

    AvailabilityZone selectedZone = zonesList.get(index);
    logger.info("You selected: " + selectedZone.zoneId());
    return selectedZone;
}

/**

```

```

    * Asynchronously sets up a new VPC, including creating the VPC, finding the
    associated route table, and
    * creating a VPC endpoint for the S3 service.
    *
    * @return a {@link CompletableFuture} that, when completed, contains a
    AbstractMap with the
    *         VPC ID and VPC endpoint ID.
    */
    public CompletableFuture<AbstractMap.SimpleEntry<String, String>>
    setupVPCAsync() {
        String cidr = "10.0.0.0/16";
        CreateVpcRequest vpcRequest = CreateVpcRequest.builder()
            .cidrBlock(cidr)
            .build();

        return getEc2AsyncClient().createVpc(vpcRequest)
            .thenCompose(vpcResponse -> {
                String vpcId = vpcResponse.vpc().vpcId();
                logger.info("VPC Created: {}", vpcId);

                Ec2AsyncWaiter waiter = getEc2AsyncClient().waiter();
                DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                    .vpcIds(vpcId)
                    .build();

                return waiter.waitUntilVpcAvailable(request)
                    .thenApply(waiterResponse -> vpcId);
            })
            .thenCompose(vpcId -> {
                Filter filter = Filter.builder()
                    .name("vpc-id")
                    .values(vpcId)
                    .build();

                DescribeRouteTablesRequest describeRouteTablesRequest =
                DescribeRouteTablesRequest.builder()
                    .filters(filter)
                    .build();

                return
                getEc2AsyncClient().describeRouteTables(describeRouteTablesRequest)
                    .thenApply(routeTablesResponse -> {
                        if (routeTablesResponse.routeTables().isEmpty()) {

```

```

        throw new CompletionException("No route tables found
for VPC: " + vpcId, null);
    }
    String routeTableId =
routeTablesResponse.routeTables().get(0).routeTableId();
    logger.info("Route table found: {}", routeTableId);
    return new AbstractMap.SimpleEntry<>(vpcId,
routeTableId);
    });
    })
    .thenCompose(vpcAndRouteTable -> {
        String vpcId = vpcAndRouteTable.getKey();
        String routeTableId = vpcAndRouteTable.getValue();
        Region region =
getEc2AsyncClient().serviceClientConfiguration().region();
        String serviceName = String.format("com.amazonaws.%s.s3express",
region.id());

        CreateVpcEndpointRequest endpointRequest =
CreateVpcEndpointRequest.builder()
            .vpcId(vpcId)
            .routeTableIds(routeTableId)
            .serviceName(serviceName)
            .build();

        return getEc2AsyncClient().createVpcEndpoint(endpointRequest)
            .thenApply(vpcEndpointResponse -> {
                String vpcEndpointId =
vpcEndpointResponse.vpcEndpoint().vpcEndpointId();
                logger.info("VPC Endpoint created: {}", vpcEndpointId);
                return new AbstractMap.SimpleEntry<>(vpcId,
vpcEndpointId);
            });
    })
    .exceptionally(exception -> {
        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
        if (cause instanceof Ec2Exception) {
            logger.error("EC2 error during VPC setup: {}",
cause.getMessage(), cause);
            throw new CompletionException("EC2 error during VPC setup: "
+ cause.getMessage(), cause);
        }
    })

```

```
        logger.error("VPC setup failed: {}", cause.getMessage(), cause);
        throw new CompletionException("VPC setup failed: " +
            cause.getMessage(), cause);
    });
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK for Java 2.x .

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObject](#)
- [GetObject](#)
- [ListObjects](#)
- [PutObject](#)

PHP

SDK per PHP

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegue uno scenario che descrive le nozioni di base relative ai bucket di directory Amazon S3 e S3 Express One Zone.

```
echo "\n";
echo "-----\n";
echo "Welcome to the Amazon S3 Express Basics demo using PHP!\n";
echo "-----\n";
```

```

    // Change these both of these values to use a different region/
availability zone.
    $region = "us-west-2";
    $az = "usw2-az1";

    $this->s3Service = new S3Service(new S3Client(['region' => $region]));
    $this->iamService = new IAMService(new IamClient(['region' => $region]));

    $uuid = uniqid();

    echo <<<INTRO
Let's get started! First, please note that S3 Express One Zone works best when
working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results
in this example, and when you implement
Directory buckets into your infrastructure, it is best to put your Compute
resources in the same AZ as your Directory
bucket.\n
INTRO;

    pressEnter();
    // 1. Configure a gateway VPC endpoint. This is the recommended method to
allow S3 Express One Zone traffic without
    // the need to pass through an internet gateway or NAT device.
    echo "\n";
    echo "1. First, we'll set up a new VPC and VPC Endpoint if this program
is running in an EC2 instance in the same AZ as your Directory buckets will be.
\n";

    $ec2Choice = testable_readline("Are you running this in an EC2 instance
located in the same AZ as your intended Directory buckets? Enter Y/y to setup a
VPC Endpoint, or N/n/blank to skip this section.");
    if($ec2Choice == "Y" || $ec2Choice == "y") {
        echo "Great! Let's set up a VPC, retrieve the Route Table from it,
and create a VPC Endpoint to connect the S3 Client to.\n";
        pressEnter();
        $this->ec2Service = new EC2Service(new Ec2Client(['region' =>
$region]));
        $cidr = "10.0.0.0/16";
        $vpc = $this->ec2Service->createVpc($cidr);
        $this->resources['vpcId'] = $vpc['VpcId'];

        $this->ec2Service->waitForVpcAvailable($vpc['VpcId']);

        $routeTable = $this->ec2Service->describeRouteTables([], [
            [

```

```

        'Name' => "vpc-id",
        'Values' => [$vpc['VpcId']],
    ],
]);

    $serviceName = "com.amazonaws." . $this->ec2Service->getRegion() .
".s3express";
    $vpcEndpoint = $this->ec2Service->createVpcEndpoint($serviceName,
    $vpc['VpcId'], [$routeTable[0]]);
    $this->resources['vpcEndpointId'] = $vpcEndpoint['VpcEndpointId'];
} else {
    echo "Skipping the VPC setup. Don't forget to use this in production!
\n";
}

// 2. Policies, user, and roles with CDK.
echo "\n";
echo "2. Policies, users, and roles with CDK.\n";
echo "Now, we'll set up some policies, roles, and a user. This user will
only have permissions to do S3 Express One Zone actions.\n";
pressEnter();

$this->cloudFormationClient = new CloudFormationClient([]);
$stackName = "cfn-stack-s3-express-basics-" . uniqid();
$file = file_get_contents(__DIR__ . "/../../../../../resources/cfn/
s3_express_basics/s3_express_template.yml");
$result = $this->cloudFormationClient->createStack([
    'StackName' => $stackName,
    'TemplateBody' => $file,
    'Capabilities' => ['CAPABILITY_IAM'],
]);
$waiter = $this->cloudFormationClient->getWaiter("StackCreateComplete",
['StackName' => $stackName]);
try {
    $waiter->promise()->wait();
} catch (CloudFormationException $caught) {
    echo "Error waiting for the CloudFormation stack to create: {$caught-
>getAwsErrorMessage()}\n";
    throw $caught;
}
$this->resources['stackName'] = $stackName;
$stackInfo = $this->cloudFormationClient->describeStacks([
    'StackName' => $result['StackId'],
]);

```



```
$expressUserName = "";
$regularUserName = "";
foreach($stackInfo['Stacks'][0]['Outputs'] as $output) {
    if ($output['OutputKey'] == "RegularUser") {
        $regularUserName = $output['OutputValue'];
    }
    if ($output['OutputKey'] == "ExpressUser") {
        $expressUserName = $output['OutputValue'];
    }
}
$regularKey = $this->iamService->createAccessKey($regularUserName);
$regularCredentials = new Credentials($regularKey['AccessKeyId'],
$regularKey['SecretAccessKey']);
$expressKey = $this->iamService->createAccessKey($expressUserName);
$expressCredentials = new Credentials($expressKey['AccessKeyId'],
$expressKey['SecretAccessKey']);

// 3. Create an additional client using the credentials with S3 Express
permissions.
echo "\n";
echo "3. Create an additional client using the credentials with S3
Express permissions.\n";
echo "This client is created with the credentials associated with the
user account with the S3 Express policy attached, so it can perform S3 Express
operations.\n";
pressEnter();
$s3RegularClient = new S3Client([
    'Region' => $region,
    'Credentials' => $regularCredentials,
]);
$s3RegularService = new S3Service($s3RegularClient);
$s3ExpressClient = new S3Client([
    'Region' => $region,
    'Credentials' => $expressCredentials,
]);
$s3ExpressService = new S3Service($s3ExpressClient);
echo "All the roles and policies were created an attached to the user.
Then, a new S3 Client and Service were created using that user's credentials.
\n";
echo "We can now use this client to make calls to S3 Express operations.
Keeping permissions in mind (and adhering to least-privilege) is crucial to S3
Express.\n";
pressEnter();
```

```
// 4. Create two buckets.
echo "\n";
echo "3. Create two buckets.\n";
echo "Now we will create a Directory bucket, which is the linchpin of the
S3 Express One Zone service.\n";
echo "Directory buckets behave in different ways from regular S3 buckets,
which we will explore here.\n";
echo "We'll also create a normal bucket, put an object into the normal
bucket, and copy it over to the Directory bucket.\n";
pressEnter();

// Create a directory bucket. These are different from normal S3 buckets
in subtle ways.
$directoryBucketName = "s3-express-demo-directory-bucket-$uuid--$az--x-
s3";
echo "Now, let's create the actual Directory bucket, as well as a regular
bucket.\n";
pressEnter();
$s3ExpressService->createBucket($directoryBucketName, [
    'CreateBucketConfiguration' => [
        'Bucket' => [
            'Type' => "Directory", // This is what causes S3 to create a
Directory bucket as opposed to a normal bucket.
            'DataRedundancy' => "SingleAvailabilityZone",
        ],
        'Location' => [
            'Name' => $az,
            'Type' => "AvailabilityZone",
        ],
    ],
]);
$this->resources['directoryBucketName'] = $directoryBucketName;

// Create a normal bucket.
$normalBucketName = "normal-bucket-$uuid";
$s3RegularService->createBucket($normalBucketName);
$this->resources['normalBucketName'] = $normalBucketName;
echo "Great! Both buckets were created.\n";
pressEnter();

// 5. Create an object and copy it over.
echo "\n";
echo "5. Create an object and copy it over.\n";
```

```
    echo "We'll create a basic object consisting of some text and upload it
to the normal bucket.\n";
    echo "Next, we'll copy the object into the Directory bucket using the
regular client.\n";
    echo "This works fine, because Copy operations are not restricted for
Directory buckets.\n";
    pressEnter();

    $objectKey = "basic-text-object";
    $s3RegularService->putObject($normalBucketName, $objectKey, $args =
['Body' => "Look Ma, I'm a bucket!"]);
    $this->resources['objectKey'] = $objectKey;

    // Create a session to access the directory bucket. The SDK Client will
automatically refresh this as needed.
    $s3ExpressService->createSession($directoryBucketName);
    $s3ExpressService->copyObject($directoryBucketName, $objectKey,
"$normalBucketName/$objectKey");

    echo "It worked! It's important to remember the user permissions when
interacting with Directory buckets.\n";
    echo "Instead of validating permissions on every call as normal buckets
do, Directory buckets utilize the user credentials and session token to
validate.\n";
    echo "This allows for much faster connection speeds on every call. For
single calls, this is low, but for many concurrent calls, this adds up to a lot
of time saved.\n";
    pressEnter();

    // 6. Demonstrate performance difference.
    echo "\n";
    echo "6. Demonstrate performance difference.\n";
    $downloads = 1000;
    echo "Now, let's do a performance test. We'll download the same object
from each bucket $downloads times and compare the total time needed. Note: the
performance difference will be much more pronounced if this example is run in an
EC2 instance in the same AZ as the bucket.\n";
    $downloadChoice = testable_readline("If you would like to download each
object $downloads times, press enter. Otherwise, enter a custom amount and press
enter.");
    if($downloadChoice && is_numeric($downloadChoice) && $downloadChoice <
1000000){ // A million is enough. I promise.
        $downloads = $downloadChoice;
    }
}
```

```
// Download the object $downloads times from each bucket and time it to
demonstrate the speed difference.
$directoryStartTime = hrtime(true);
for($i = 0; $i < $downloads; ++$i){
    $s3ExpressService->getObject($directoryBucketName, $objectKey);
}
$directoryEndTime = hrtime(true);
$directoryTimeDiff = $directoryEndTime - $directoryStartTime;

$normalStartTime = hrtime(true);
for($i = 0; $i < $downloads; ++$i){
    $s3RegularService->getObject($normalBucketName, $objectKey);
}
$normalEndTime = hrtime(true);
$normalTimeDiff = $normalEndTime - $normalStartTime;

echo "The directory bucket took $directoryTimeDiff nanoseconds, while the
normal bucket took $normalTimeDiff.\n";
echo "That's a difference of " . ($normalTimeDiff - $directoryTimeDiff) .
" nanoseconds, or " . (($normalTimeDiff - $directoryTimeDiff)/1000000000) . "
seconds.\n";
pressEnter();

// 7. Populate the buckets to show the lexicographical difference.
echo "\n";
echo "7. Populate the buckets to show the lexicographical difference.\n";
echo "Now let's explore how Directory buckets store objects in a
different manner to regular buckets.\n";
echo "The key is in the name \"Directory!\"\n";
echo "Where regular buckets store their key/value pairs in a flat manner,
Directory buckets use actual directories/folders.\n";
echo "This allows for more rapid indexing, traversing, and therefore
retrieval times!\n";
echo "The more segmented your bucket is, with lots of directories, sub-
directories, and objects, the more efficient it becomes.\n";
echo "This structural difference also causes ListObjects to behave
differently, which can cause unexpected results.\n";
echo "Let's add a few more objects with layered directories as see how
the output of ListObjects changes.\n";
pressEnter();

// Populate a few more files in each bucket so that we can use
ListObjects and show the difference.
```

```
$otherObject = "other/$objectKey";
$altObject = "alt/$objectKey";
$otherAltObject = "other/alt/$objectKey";
$s3ExpressService->putObject($directoryBucketName, $otherObject);
$s3RegularService->putObject($normalBucketName, $otherObject);
$this->resources['otherObject'] = $otherObject;
$s3ExpressService->putObject($directoryBucketName, $altObject);
$s3RegularService->putObject($normalBucketName, $altObject);
$this->resources['altObject'] = $altObject;
$s3ExpressService->putObject($directoryBucketName, $otherAltObject);
$s3RegularService->putObject($normalBucketName, $otherAltObject);
$this->resources['otherAltObject'] = $otherAltObject;

$listDirectoryBucket = $s3ExpressService-
>listObjects($directoryBucketName);
$listNormalBucket = $s3RegularService->listObjects($normalBucketName);

// Directory bucket content
echo "Directory bucket content\n";
foreach($listDirectoryBucket['Contents'] as $result){
    echo $result['Key'] . "\n";
}

// Normal bucket content
echo "\nNormal bucket content\n";
foreach($listNormalBucket['Contents'] as $result){
    echo $result['Key'] . "\n";
}

echo "Notice how the normal bucket lists objects in lexicographical
order, while the directory bucket does not. This is because the normal bucket
considers the whole \"key\" to be the object identifies, while the directory
bucket actually creates directories and uses the object \"key\" as a path to the
object.\n";
pressEnter();

echo "\n";
echo "That's it for our tour of the basic operations for S3 Express One
Zone.\n";
$cleanUp = testable_readline("Would you like to delete all the resources
created during this demo? Enter Y/y to delete all the resources.");
if($cleanUp){
    $this->cleanUp();
}
```

```
namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
        $this->verbose = $verbose;
    }

    public function setVerbose($verbose)
    {
        $this->verbose = $verbose;
    }

    public function isVerbose(): bool
    {
        return $this->verbose;
    }

    public function getClient(): S3Client
    {
        return $this->client;
    }
}
```

```
}

public function setClient(S3Client $client)
{
    $this->client = $client;
}

public function emptyAndDeleteBucket($bucketName, array $args = [])
{
    try {
        $objects = $this->listAllObjects($bucketName, $args);
        $this->deleteObjects($bucketName, $objects, $args);
        if ($this->verbose) {
            echo "Deleted all objects and folders from $bucketName.\n";
        }
        $this->deleteBucket($bucketName, $args);
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
            echo "\nPlease fix error with bucket deletion before continuing.
\n";
        }
        throw $exception;
    }
}

public function createBucket(string $bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);
    try {
        $this->client->createBucket($parameters);
        if ($this->verbose) {
            echo "Created the bucket named: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket creation before continuing.";
        }
    }
}
```

```

        throw $exception;
    }
}

public function putObject(string $bucketName, string $key, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
    $args);
    try {
        $this->client->putObject($parameters);
        if ($this->verbose) {
            echo "Uploaded the object named: $key to the bucket named:
$bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create $key in $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object uploading before continuing.";
        }
        throw $exception;
    }
}

public function getObject(string $bucketName, string $key, array $args = []):
Result
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
    $args);
    try {
        $object = $this->client->getObject($parameters);
        if ($this->verbose) {
            echo "Downloaded the object named: $key to the bucket named:
$bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to download $key from $bucketName with error:
{$exception->getMessage()}\n";

```



```
        echo "Please fix error with object downloading before
continuing.";
    }
    throw $exception;
}
return $object;
}

public function copyObject($bucketName, $key, $copySource, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
"CopySource" => $copySource], $args);
    try {
        $this->client->copyObject($parameters);
        if ($this->verbose) {
            echo "Copied the object from: $copySource in $bucketName to:
$key.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to copy $copySource in $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object copying before continuing.";
        }
        throw $exception;
    }
}

public function listObjects(string $bucketName, $start = 0, $max = 1000,
array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
    try {
        $objects = $this->client->listObjectsV2($parameters);
        if ($this->verbose) {
            echo "Retrieved the list of objects from: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
```

```
        echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
        echo "Please fix error with list objects before continuing.";
    }
    throw $exception;
}
return $objects;
}

public function listAllObjects($bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);

    $contents = [];
    $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

    foreach ($paginator as $result) {
        if($result['KeyCount'] == 0){
            break;
        }
        foreach ($result['Contents'] as $object) {
            $contents[] = $object;
        }
    }
    return $contents;
}

public function deleteObjects(string $bucketName, array $objects, array $args
= [])
{
    $listOfObjects = array_map(
        function ($object) {
            return ['Key' => $object];
        },
        array_column($objects, 'Key')
    );
    if(!$listOfObjects){
        return;
    }
}
```

```

        $parameters = array_merge(['Bucket' => $bucketName, 'Delete' =>
['Objects' => $listOfObjects]], $args);
        try {
            $this->client->deleteObjects($parameters);
            if ($this->verbose) {
                echo "Deleted the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }

    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket deletion before continuing.";
            }
            throw $exception;
        }
    }

    public function deleteObject(string $bucketName, string $fileName, array
$args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
$args);

```

```

        try {
            $this->client->deleteObject($parameters);
            if ($this->verbose) {
                echo "Deleted the object named: $fileName from $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }

    public function listBuckets(array $args = [])
    {
        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {
                echo "Retrieved all " . count($buckets) . "\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket lists before continuing.";
            }
            throw $exception;
        }
        return $buckets;
    }

    public function preSignedUrl(CommandInterface $command, DateTimeInterface|
int|string $expires, array $options = [])
    {
        $request = $this->client->createPresignedRequest($command, $expires,
$options);
        try {
            $presignedUrl = (string)$request->getUri();

```

```
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
                echo "Please fix error with presigned urls before continuing.";
            }
            throw $exception;
        }
        return $presignedUrl;
    }

    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
                'Bucket' => $bucketName,
            ]);
            return $result;
        }catch(S3Exception $caught){
            if($caught->getAwsErrorType() == "NoSuchBucket"){
                echo "The specified bucket does not exist.";
            }
            throw $caught;
        }
    }
}
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per PHP .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)
 - [ListObjects](#)

- [PutObject](#)

Python

SDK per Python (Boto3)

Note

C'è dell'altro GitHub. Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Esegue uno scenario che descrive le nozioni di base relative ai bucket di directory Amazon S3 e S3 Express One Zone.

```
class S3ExpressScenario:
    """Runs an interactive scenario that shows how to get started with S3
    Express."""

    def __init__(
        self,
        cloud_formation_resource: ServiceResource,
        ec2_client: client,
        iam_client: client,
    ):
        self.cloud_formation_resource = cloud_formation_resource
        self.ec2_client = ec2_client
        self.iam_client = iam_client
        self.region = ec2_client.meta.region_name
        self.stack = None
        self.vpc_id = None
        self.vpc_endpoint_id = None
        self.regular_bucket_name = None
        self.directory_bucket_name = None
        self.s3_express_wrapper = None
        self.s3_regular_wrapper = None

    def s3_express_scenario(self):
        """
        Runs the scenario.
        """
```

```

        print("")
        print_dashes()
        print("Welcome to the Amazon S3 Express Basics demo using Python (Boto
3)!")
        print_dashes()
        print(
            """

```

Let's get started! First, please note that S3 Express One Zone works best when working within the AWS infrastructure, specifically when working in the same Availability Zone. To see the best results in this example and when you implement Directory buckets into your infrastructure, it is best to put your compute resources in the same AZ as your Directory bucket.

```

        """
    )
    press_enter_to_continue()

    # Create an optional VPC and create 2 IAM users.
    express_user_name, regular_user_name = self.create_vpc_and_users()

    # Set up two S3 clients, one regular and one express, and two buckets,
    one regular and one express.
    self.setup_clients_and_buckets(express_user_name, regular_user_name)

    # Create an S3 session for the express S3 client and add objects to the
    buckets.
    bucket_object = self.create_session_and_add_objects()

    # Demonstrate performance differences between regular and express
    buckets.
    self.demonstrate_performance(bucket_object)

    # Populate the buckets to show the lexicographical difference between
    regular and express buckets.
    self.show_lexicographical_differences(bucket_object)

    print("")
    print("That's it for our tour of the basic operations for S3 Express One
Zone.")

    if q.ask(
        "Would you like to delete all the resources created during this demo
(y/n)? ",

```

```

        q.is_ynsno,
    ):
        self.cleanup()

    def create_vpc_and_users(self) -> None:
        """
        Optionally create a VPC.
        Create two IAM users, one with S3 Express One Zone permissions and one
        without.
        """
        # Configure a gateway VPC endpoint. This is the recommended method to
        allow S3 Express One Zone traffic without
        # the need to pass through an internet gateway or NAT device.
        print(
            """
1. First, we'll set up a new VPC and VPC Endpoint if this program is running in
   an EC2 instance in the same AZ as your
Directory buckets will be. Are you running this in an EC2 instance located in the
   same AZ as your intended Directory buckets?
"""
        )
        if q.ask("Do you want to setup a VPC Endpoint? (y/n) ", q.is_ynsno):
            print(
                "Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to."
            )
            self.setup_vpc()
            press_enter_to_continue()
        else:
            print("Skipping the VPC setup. Don't forget to use this in
production!")
            print(
                """
2. Policies, users, and roles with CDK.
Now, we'll set up some policies, roles, and a user. This user will only have
permissions to do S3 Express One Zone actions.
"""
            )
            press_enter_to_continue()
            stack_name = f"cfn-stack-s3-express-basics--{uuid.uuid4()}"
            template_as_string = S3ExpressScenario.get_template_as_string()
            self.stack = self.deploy_cloudformation_stack(stack_name,
template_as_string)
            regular_user_name = None

```



```

        express_user_name = None
        outputs = self.stack.outputs
        for output in outputs:
            if output.get("OutputKey") == "RegularUser":
                regular_user_name = output.get("OutputValue")
            elif output.get("OutputKey") == "ExpressUser":
                express_user_name = output.get("OutputValue")
        if not regular_user_name or not express_user_name:
            error_string = f"""
            Failed to retrieve required outputs from CloudFormation stack.
            'regular_user_name'={regular_user_name},
            'express_user_name'={express_user_name}
            """
            logger.error(error_string)
            raise ValueError(error_string)
        return express_user_name, regular_user_name

    def setup_clients_and_buckets(
        self, express_user_name: str, regular_user_name: str
    ) -> None:
        """
        Set up two S3 clients, one regular and one express, and two buckets, one
        regular and one express.
        :param express_user_name: The name of the user with S3 Express
        permissions.
        :param regular_user_name: The name of the user with regular S3
        permissions.
        """
        regular_credentials = self.create_access_key(regular_user_name)
        express_credentials = self.create_access_key(express_user_name)
        # 3. Create an additional client using the credentials with S3 Express
        permissions.
        print(
            """
3. Create an additional client using the credentials with S3 Express permissions.
This client is created with the
credentials associated with the user account with the S3 Express policy attached,
so it can perform S3 Express operations.
"""
        )
        press_enter_to_continue()
        s3_regular_client = self.create_s3_client_with_access_key_credentials(
            regular_credentials
        )

```

```

self.s3_regular_wrapper = S3ExpressWrapper(s3_regular_client)
s3_express_client = self.create_s3__client_with_access_key_credentials(
    express_credentials
)
self.s3_express_wrapper = S3ExpressWrapper(s3_express_client)
print(
    """

```

All the roles and policies were created and attached to the user. Then a new S3 Client were created using that user's credentials. We can now use this client to make calls to S3 Express operations. Keeping permissions in mind (and adhering to least-privilege) is crucial to S3 Express.

```

    """
)
press_enter_to_continue()
# 4. Create two buckets.
print(
    """

```

3. Create two buckets.

Now we will create a Directory bucket which is the linchpin of the S3 Express One Zone service. Directory buckets

behave in different ways from regular S3 buckets which we will explore here.

We'll also create a normal bucket, put an object into the normal bucket, and copy it over to the Directory bucket.

```

    """
)

# Create a directory bucket. These are different from normal S3 buckets
in subtle ways.
bucket_prefix = q.ask(
    "Enter a bucket name prefix that will be used for both buckets: ",
    q.re_match(r"[a-z0-9](?:[a-z0-9-\.]*)[a-z0-9]$"),
)

# Some availability zones are not supported for Directory buckets. We'll
choose one that is supported.
print(
    "Now, let's choose an availability zone for the Directory bucket.
We'll choose one that is supported."
)
while True:
    availability_zone = self.select_availability_zone_id(self.region)
    # Construct the parts of a directory bucket name that is made unique
with a UUID string.

```

```

        directory_bucket_suffix = f"--{availability_zone['ZoneId']}--x-s3"
        max_uuid_length = 63 - len(bucket_prefix) -
len(directory_bucket_suffix) - 1
        bucket_uuid = str(uuid.uuid4()).replace("-", "")[:max_uuid_length]
        directory_bucket_name = (
            f"{bucket_prefix}-{bucket_uuid}{directory_bucket_suffix}"
        )
        regular_bucket_name = f"{bucket_prefix}-regular-{bucket_uuid}"
        configuration = {
            "Bucket": {
                "Type": "Directory",
                "DataRedundancy": "SingleAvailabilityZone",
            },
            "Location": {
                "Name": availability_zone["ZoneId"],
                "Type": "AvailabilityZone",
            },
        }
        press_enter_to_continue()
        print(
            "Now, let's create the actual Directory bucket, as well as a
regular bucket."
        )
        press_enter_to_continue()
        try:
            self.s3_express_wrapper.create_bucket(
                directory_bucket_name, configuration
            )
            break
        except ClientError as client_error:
            if client_error.response["Error"]["Code"] == "InvalidBucketName":
                print(
                    f"Bucket '{directory_bucket_name}' is invalid. This may
be because of selected availability zone."
                )
                if q.ask(
                    "Would you like to select a different availability zone?"
                ),
                    q.is_yesno,
                ):
                    continue
                else:
                    raise
            else:

```

```

        raise

    print(f"Created directory bucket, '{directory_bucket_name}'")
    self.directory_bucket_name = directory_bucket_name

    self.s3_regular_wrapper.create_bucket(regular_bucket_name)
    print(f"Created regular bucket, '{regular_bucket_name}'")
    self.regular_bucket_name = regular_bucket_name
    print("Great! Both buckets were created.")
    press_enter_to_continue()

def create_session_and_add_objects(self) -> None:
    """
    Create a session for the express S3 client and add objects to the
    buckets.
    """
    print(
        """
5. Create an object and copy it over.
We'll create a basic object consisting of some text and upload it to the normal
bucket. Next we'll copy the object
into the Directory bucket using the regular client. This works fine because copy
operations are not restricted for
Directory buckets.
        """
    )
    press_enter_to_continue()
    bucket_object = "basic-text-object"
    self.s3_regular_wrapper.put_object(
        self.regular_bucket_name, bucket_object, "Look Ma, I'm a bucket!"
    )
    self.s3_express_wrapper.create_session(self.directory_bucket_name)
    self.s3_express_wrapper.copy_object(
        self.regular_bucket_name,
        bucket_object,
        self.directory_bucket_name,
        bucket_object,
    )
    print(
        """

```

It worked! It's important to remember the user permissions when interacting with Directory buckets. Instead of validating permissions on every call as normal buckets do, Directory buckets utilize the user credentials and session token to validate.

This allows for much faster connection speeds on every call. For single calls, this is low, but for many concurrent calls this adds up to a lot of time saved.

```
"""
```

```
    )
    press_enter_to_continue()
    return bucket_object
```

```
def demonstrate_performance(self, bucket_object: str) -> None:
```

```
    """
```

```
    Demonstrate performance differences between regular and Directory
    buckets.
```

```
    :param bucket_object: The name of the object to download from each
    bucket.
```

```
    """
```

```
    print("")
    print("6. Demonstrate performance difference.")
    print(
        """
```

Now, let's do a performance test. We'll download the same object from each bucket 'downloads' times

and compare the total time needed. Note: the performance difference will be much more pronounced if this

example is run in an EC2 instance in the same Availability Zone as the bucket.

```
"""
```

```
    )
    downloads = 1000
```

```
    print(
        f"The number of downloads of the same object for this example is set
        at {downloads}."
    )
```

```
    if q.ask("Would you like to download a different number? (y/n) ",
    q.is_yesno):
```

```
        max_downloads = 1000000
```

```
        downloads = q.ask(
```

```
            f"Enter a number between 1 and {max_downloads} for the number of
            downloads: ",
```

```
            q.is_int,
            q.in_range(1, max_downloads),
        )
```

```
        # Download the object 'downloads' times from each bucket and time it to
        demonstrate the speed difference.
```

```
        print("Downloading from the Directory bucket.")
```

```
        directory_time_start = time.time_ns()
```

```

for index in range(downloads):
    if index % 10 == 0:
        print(f"Download {index} of {downloads}")

    self.s3_express_wrapper.get_object(
        self.directory_bucket_name, bucket_object
    )

directory_time_difference = time.time_ns() - directory_time_start
print("Downloading from the normal bucket.")
normal_time_start = time.time_ns()

for index in range(downloads):
    if index % 10 == 0:
        print(f"Download {index} of {downloads}")
    self.s3_regular_wrapper.get_object(self.regular_bucket_name,
bucket_object)

normal_time_difference = time.time_ns() - normal_time_start
print(
    f"The directory bucket took {directory_time_difference} nanoseconds,
while the normal bucket took {normal_time_difference}."
)
difference = normal_time_difference - directory_time_difference
print(f"That's a difference of {difference} nanoseconds, or")
print(f"{{(difference) / 1000000000}} seconds.")
if difference < 0:
    print(
        "The directory buckets were slower. This can happen if you are
not running on the cloud within a vpc."
    )
    press_enter_to_continue()

def show_lexicographical_differences(self, bucket_object: str) -> None:
    """
    Show the lexicographical difference between Directory buckets and regular
    buckets.
    This is done by creating a few objects in each bucket and listing them to
    show the difference.
    :param bucket_object: The object to use for the listing operations.
    """
    print(
        """

```

7. Populate the buckets to show the lexicographical difference.

Now let's explore how Directory buckets store objects in a different manner to regular buckets. The key is in the name "Directory". Where regular buckets store their key/value pairs in a flat manner, Directory buckets use actual directories/folders. This allows for more rapid indexing, traversing, and therefore retrieval times! The more segmented your bucket is, with lots of directories, sub-directories, and objects, the more efficient it becomes. This structural difference also causes ListObjects to behave differently, which can cause unexpected results. Let's add a few more objects with layered directories to see how the output of ListObjects changes.

```

"""
)
press_enter_to_continue()
# Populate a few more files in each bucket so that we can use ListObjects
and show the difference.
other_object = f"other/{bucket_object}"
alt_object = f"alt/{bucket_object}"
other_alt_object = f"other/alt/{bucket_object}"
self.s3_regular_wrapper.put_object(self.regular_bucket_name,
other_object, "")
self.s3_express_wrapper.put_object(self.directory_bucket_name,
other_object, "")
self.s3_regular_wrapper.put_object(self.regular_bucket_name, alt_object,
"")
self.s3_express_wrapper.put_object(self.directory_bucket_name,
alt_object, "")
self.s3_regular_wrapper.put_object(
    self.regular_bucket_name, other_alt_object, ""
)
self.s3_express_wrapper.put_object(
    self.directory_bucket_name, other_alt_object, ""
)
directory_bucket_objects = self.s3_express_wrapper.list_objects(
    self.directory_bucket_name
)

regular_bucket_objects = self.s3_regular_wrapper.list_objects(
    self.regular_bucket_name
)

print("Directory bucket content")
for bucket_object in directory_bucket_objects:

```

```

        print(f"    {bucket_object['Key']}")
    print("Normal bucket content")
    for bucket_object in regular_bucket_objects:
        print(f"    {bucket_object['Key']}")
    print(
        """
Notice how the normal bucket lists objects in lexicographical order, while the
directory bucket does not. This is
because the normal bucket considers the whole "key" to be the object identifier,
while the directory bucket actually
creates directories and uses the object "key" as a path to the object.
        """
    )
    press_enter_to_continue()

def cleanup(self) -> None:
    """
    Delete resources created by this scenario.
    """
    if self.directory_bucket_name is not None:
        self.s3_express_wrapper.delete_bucket_and_objects(
            self.directory_bucket_name
        )
        print(f"Deleted directory bucket, '{self.directory_bucket_name}'")
        self.directory_bucket_name = None

    if self.regular_bucket_name is not None:

self.s3_regular_wrapper.delete_bucket_and_objects(self.regular_bucket_name)
        print(f"Deleted regular bucket, '{self.regular_bucket_name}'")
        self.regular_bucket_name = None

    if self.stack is not None:
        self.destroy_cloudformation_stack(self.stack)
        self.stack = None

    self.tear_done_vpc()

def create_access_key(self, user_name: str) -> dict[str, any]:
    """
    Creates an access key for the user.
    :param user_name: The name of the user.
    :return: The access key for the user.
    """

```



```

    try:
        access_key = self.iam_client.create_access_key(UserName=user_name)
        return access_key["AccessKey"]
    except ClientError as client_error:
        logging.error(
            "Couldn't create the access key. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise

def create_s3_client_with_access_key_credentials(
    self, access_key: dict[str, any]
) -> client:
    """
    Creates an S3 client with access key credentials.
    :param access_key: The access key for the user.
    :return: The S3 Express One Zone client.
    """
    try:
        s3_express_client = boto3.client(
            "s3",
            aws_access_key_id=access_key["AccessKeyId"],
            aws_secret_access_key=access_key["SecretAccessKey"],
            region_name=self.region,
        )
        return s3_express_client
    except ClientError as client_error:
        logging.error(
            "Couldn't create the S3 Express One Zone client. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise

def select_availability_zone_id(self, region: str) -> dict[str, any]:
    """
    Selects an availability zone.
    :param region: The region to select the availability zone from.
    :return: The availability zone dictionary.
    """
    try:
        response = self.ec2_client.describe_availability_zones(
            Filters=[{"Name": "region-name", "Values": [region]}]
        )
        availability_zones = response["AvailabilityZones"]

```

```

        zone_names = [zone["ZoneName"] for zone in availability_zones]
        index = q.choose("Select an availability zone: ", zone_names)
        return availability_zones[index]
    except ClientError as client_error:
        logging.error(
            "Couldn't describe availability zones. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise

def deploy_cloudformation_stack(
    self, stack_name: str, cfn_template: str
) -> ServiceResource:
    """
    Deploys prerequisite resources used by the scenario. The resources are
    defined in the associated `cfn_template.yaml` AWS CloudFormation script
and are deployed
    as a CloudFormation stack, so they can be easily managed and destroyed.

    :param stack_name: The name of the CloudFormation stack.
    :param cfn_template: The CloudFormation template as a string.
    :return: The CloudFormation stack resource.
    """
    print(f"Deploying CloudFormation stack: {stack_name}.")
    stack = self.cloud_formation_resource.create_stack(
        StackName=stack_name,
        TemplateBody=cfn_template,
        Capabilities=["CAPABILITY_NAMED_IAM"],
    )
    print(f"CloudFormation stack creation started: {stack_name}")
    print("Waiting for CloudFormation stack creation to complete...")
    waiter = self.cloud_formation_resource.meta.client.get_waiter(
        "stack_create_complete"
    )
    waiter.wait(StackName=stack.name)
    stack.load()
    print("CloudFormation stack creation complete.")

    return stack

def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
    """
    Destroys the resources managed by the CloudFormation stack, and the
CloudFormation

```

```

        stack itself.

        :param stack: The CloudFormation stack that manages the example
resources.
        """
        try:
            print(
                f"CloudFormation stack '{stack.name}' is being deleted. This may
take a few minutes."
            )
            stack.delete()
            waiter = self.cloud_formation_resource.meta.client.get_waiter(
                "stack_delete_complete"
            )
            waiter.wait(StackName=stack.name)
            print(f"CloudFormation stack '{stack.name}' has been deleted.")
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the CloudFormation stack. Here's why: %s",
                client_error.response["Error"]["Message"],
            )

    @staticmethod
    def get_template_as_string() -> str:
        """
        Returns a string containing this scenario's CloudFormation template.
        """
        script_directory = os.path.dirname(os.path.abspath(__file__))
        template_file_path = os.path.join(script_directory,
"s3_express_template.yaml")
        file = open(template_file_path, "r")
        return file.read()

    def setup_vpc(self):
        cidr = "10.0.0.0/16"
        try:
            response = self.ec2_client.create_vpc(CidrBlock=cidr)
            self.vpc_id = response["Vpc"]["VpcId"]

            waiter = self.ec2_client.get_waiter("vpc_available")
            waiter.wait(VpcIds=[self.vpc_id])
            print(f"Created vpc {self.vpc_id}")

        except ClientError as client_error:

```

```

        logging.error(
            "Couldn't create the vpc. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise
    try:
        response = self.ec2_client.describe_route_tables(
            Filters=[{"Name": "vpc-id", "Values": [self.vpc_id]}]
        )
        route_table_id = response["RouteTables"][0]["RouteTableId"]
        service_name = f"com.amazonaws.
{self.ec2_client.meta.region_name}.s3express"

        response = self.ec2_client.create_vpc_endpoint(
            VpcId=self.vpc_id,
            RouteTableIds=[route_table_id],
            ServiceName=service_name,
        )
        self.vpc_endpoint_id = response["VpcEndpoint"]["VpcEndpointId"]
        print(f"Created vpc endpoint {self.vpc_endpoint_id}")

    except ClientError as client_error:
        logging.error(
            "Couldn't create the vpc endpoint. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise

def tear_down_vpc(self) -> None:
    if self.vpc_endpoint_id is not None:
        try:
            self.ec2_client.delete_vpc_endpoints(
                VpcEndpointIds=[self.vpc_endpoint_id]
            )
            print(f"Deleted vpc endpoint {self.vpc_endpoint_id}.")
            self.vpc_endpoint_id = None
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the vpc endpoint %s. Here's why: %s",
                self.vpc_endpoint_id,
                client_error.response["Error"]["Message"],
            )
    if self.vpc_id is not None:
        try:

```

```

        self.ec2_client.delete_vpc(VpcId=self.vpc_id)
        print(f"Deleted vpc {self.vpc_id}")
        self.vpc_id = None
    except ClientError as client_error:
        logging.error(
            "Couldn't delete the vpc %s. Here's why: %s",
            self.vpc_id,
            client_error.response["Error"]["Message"],
        )

```

Una classe wrapper per le funzioni dell'SDK di Amazon S3 Express.

```

class S3ExpressWrapper:
    """Encapsulates Amazon S3 Express One Zone actions using the client
    interface."""

    def __init__(self, s3_client: Any) -> None:
        """
        Initializes the S3ExpressWrapper with an S3 client.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                           access to AWS S3 services.
        """
        self.s3_client = s3_client

    @classmethod
    def from_client(cls) -> "S3ExpressWrapper":
        """
        Creates an S3ExpressWrapper instance with a default s3 client.

        :return: An instance of S3ExpressWrapper initialized with the default S3
client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def create_bucket(
        self, bucket_name: str, bucket_configuration: dict[str, any] = None

```

```

) -> None:
    """
    Creates a bucket.
    :param bucket_name: The name of the bucket.
    :param bucket_configuration: The optional configuration for the bucket.
    """
    try:
        params = {"Bucket": bucket_name}
        if bucket_configuration:
            params["CreateBucketConfiguration"] = bucket_configuration

        self.s3_client.create_bucket(**params)
    except ClientError as client_error:
        # Do not log InvalidBucketName error because it is logged elsewhere.
        if client_error.response["Error"]["Code"] != "InvalidBucketName":
            logging.error(
                "Couldn't create the bucket %s. Here's why: %s",
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise

def delete_bucket_and_objects(self, bucket_name: str) -> None:
    """
    Deletes a bucket and its objects.
    :param bucket_name: The name of the bucket.
    """
    try:
        # Delete the objects in the bucket first. This is required for a
        bucket to be deleted.
        paginator = self.s3_client.get_paginator("list_objects_v2")
        page_iterator = paginator.paginate(Bucket=bucket_name)
        for page in page_iterator:
            if "Contents" in page:
                delete_keys = {
                    "Objects": [{"Key": obj["Key"]} for obj in
page["Contents"]]
                }
                response = self.s3_client.delete_objects(
                    Bucket=bucket_name, Delete=delete_keys
                )
                if "Errors" in response:
                    for error in response["Errors"]:
                        logging.error(

```

```

        "Couldn't delete object %s. Here's why: %s",
        error["Key"],
        error["Message"],
    )

    self.s3_client.delete_bucket(Bucket=bucket_name)
except ClientError as client_error:
    logging.error(
        "Couldn't delete the bucket %s. Here's why: %s",
        bucket_name,
        client_error.response["Error"]["Message"],
    )

def put_object(self, bucket_name: str, object_key: str, content: str) ->
None:
    """
    Puts an object into a bucket.
    :param bucket_name: The name of the bucket.
    :param object_key: The key of the object.
    :param content: The content of the object.
    """
    try:
        self.s3_client.put_object(Body=content, Bucket=bucket_name,
Key=object_key)
    except ClientError as client_error:
        logging.error(
            "Couldn't put the object %s into bucket %s. Here's why: %s",
            object_key,
            bucket_name,
            client_error.response["Error"]["Message"],
        )
        raise

def list_objects(self, bucket: str) -> list[str]:
    """
    Lists objects in a bucket.
    :param bucket: The name of the bucket.
    :return: The list of objects in the bucket.
    """
    try:
        response = self.s3_client.list_objects_v2(Bucket=bucket)
        return response.get("Contents", [])
    except ClientError as client_error:
        logging.error(

```

```

        "Couldn't list objects in bucket %s. Here's why: %s",
        bucket,
        client_error.response["Error"]["Message"],
    )
    raise

def copy_object(
    self,
    source_bucket: str,
    source_key: str,
    destination_bucket: str,
    destination_key: str,
) -> None:
    """
    Copies an object from one bucket to another.
    :param source_bucket: The source bucket.
    :param source_key: The source key.
    :param destination_bucket: The destination bucket.
    :param destination_key: The destination key.
    :return: None
    """
    try:
        self.s3_client.copy_object(
            CopySource={"Bucket": source_bucket, "Key": source_key},
            Bucket=destination_bucket,
            Key=destination_key,
        )
    except ClientError as client_error:
        logging.error(
            "Couldn't copy object %s from bucket %s to bucket %s. Here's why:
%s",

            source_key,
            source_bucket,
            destination_bucket,
            client_error.response["Error"]["Message"],
        )
    raise

def create_session(self, bucket_name: str) -> None:
    """
    Creates an express session.
    :param bucket_name: The name of the bucket.
    """
    try:

```



```
        self.s3_client.create_session(Bucket=bucket_name)
    except ClientError as client_error:
        logging.error(
            "Couldn't create the express session for bucket %s. Here's why:
%s",
            bucket_name,
            client_error.response["Error"]["Message"],
        )
        raise

def get_object(self, bucket_name: str, object_key: str) -> None:
    """
    Gets an object from a bucket.
    :param bucket_name: The name of the bucket.
    :param object_key: The key of the object.
    """
    try:
        self.s3_client.get_object(Bucket=bucket_name, Key=object_key)
    except ClientError as client_error:
        logging.error(
            "Couldn't get the object %s from bucket %s. Here's why: %s",
            object_key,
            bucket_name,
            client_error.response["Error"]["Message"],
        )
        raise
```

- Per informazioni dettagliate sull'API, consulta i seguenti argomenti nella documentazione di riferimento dell'API AWS SDK per Python (Boto3).
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)
 - [ListObjects](#)
 - [PutObject](#)

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Azioni per l'utilizzo di S3 Directory Bucket AWS SDK

I seguenti esempi di codice mostrano come eseguire singole azioni S3 Directory Bucket con gli SDK. AWS Ogni esempio include un collegamento a GitHub, dove puoi trovare le istruzioni per la configurazione e l'esecuzione del codice.

Questi estratti chiamano l'API dei bucket di directory di S3 Glacier e sono estratti di codice da programmi più grandi che devono essere eseguiti in modo contestuale. È possibile visualizzare le azioni nel contesto in [Scenari per l'utilizzo di S3 Directory Bucket AWS SDK](#).

Gli esempi seguenti includono solo le azioni più comunemente utilizzate. Per un elenco completo, consulta la [documentazione di riferimento dell'API per i bucket di directory S3](#).

Esempi

- [Utilizzare AbortMultipartUploadcon un AWS SDK](#)
- [Utilizzare CompleteMultipartUploadcon un AWS SDK](#)
- [Utilizzare CopyObjectcon un AWS SDK](#)
- [Utilizzare CreateBucketcon un AWS SDK](#)
- [Utilizzare CreateMultipartUploadcon un AWS SDK](#)
- [Utilizzare CreateSessioncon un AWS SDK](#)
- [Utilizzare DeleteBucketcon un AWS SDK](#)
- [Utilizzare DeleteBucketEncryptioncon un AWS SDK](#)
- [Utilizzare DeleteBucketPolicycon un AWS SDK](#)
- [Utilizzare DeleteObjectcon un AWS SDK](#)
- [Utilizzare DeleteObjectscon un AWS SDK](#)
- [Utilizzare GetBucketEncryptioncon un AWS SDK](#)
- [Utilizzare GetBucketPolicycon un AWS SDK](#)
- [Utilizzare GetObjectcon un AWS SDK](#)
- [Utilizzare GetObjectAttributescon un AWS SDK](#)
- [Utilizzare HeadBucketcon un AWS SDK](#)

- [Utilizzare HeadObject con un AWS SDK](#)
- [Utilizzare ListDirectoryBuckets con un AWS SDK](#)
- [Utilizzare ListMultipartUploads con un AWS SDK](#)
- [Usa ListObjectsV2 con un AWS SDK](#)
- [Utilizzare ListParts con un AWS SDK](#)
- [Utilizzare PutBucketEncryption con un AWS SDK](#)
- [Utilizzare PutBucketPolicy con un AWS SDK](#)
- [Utilizzare PutObject con un AWS SDK](#)
- [Utilizzare UploadPart con un AWS SDK](#)
- [Utilizzare UploadPartCopy con un AWS SDK](#)

Utilizzare **AbortMultipartUpload** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare **AbortMultipartUpload**.

Java

SDK per Java 2.x

 Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Interrompe un caricamento in più parti in un bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
```

```

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Aborts a specific multipart upload for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be uploaded
 * @param uploadId The upload ID of the multipart upload to abort
 * @return True if the multipart upload is successfully aborted, false
 otherwise
 */
public static boolean abortDirectoryBucketMultipartUpload(S3Client s3Client,
String bucketName,
String objectKey, String uploadId) {
    logger.info("Aborting multipart upload: {} for bucket: {}", uploadId,
bucketName);
    try {
        // Abort the multipart upload
        AbortMultipartUploadRequest abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .uploadId(uploadId)
            .build();

        s3Client.abortMultipartUpload(abortMultipartUploadRequest);
        logger.info("Aborted multipart upload: {} for object: {}", uploadId,
objectKey);
        return true;
    } catch (S3Exception e) {
        logger.error("Failed to abort multipart upload: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        return false;
    }
}

```

- Per i dettagli sull'API, consulta la [AbortMultipartUpload](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **CompleteMultipartUpload** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `CompleteMultipartUpload`.

Java

SDK per Java 2.x

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Completa un caricamento in più parti in un bucket di directory.

```
import com.example.s3.util.S3DirectoryBucketUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
```

```

import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * This method completes the multipart upload request by collating all the
 * upload parts.
 *
 * @param s3Client    The S3 client used to interact with S3
 * @param bucketName  The name of the directory bucket
 * @param objectKey   The key (name) of the object to be uploaded
 * @param uploadId    The upload ID used to track the multipart upload
 * @param uploadParts The list of completed parts
 * @return True if the multipart upload is successfully completed, false
 *         otherwise
 */
public static boolean completeDirectoryBucketMultipartUpload(S3Client
s3Client, String bucketName, String objectKey,
String uploadId, List<CompletedPart> uploadParts) {
    try {
        CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
            .parts(uploadParts)
            .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompleteMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .uploadId(uploadId)
            .multipartUpload(completedMultipartUpload)
            .build();

        CompleteMultipartUploadResponse response =
s3Client.completeMultipartUpload(completeMultipartUploadRequest);
        logger.info("Multipart upload completed. ETag: {}", response.eTag());
        return true;
    } catch (S3Exception e) {
        logger.error("Failed to complete multipart upload: {} - Error code:
{}", e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        return false;
    }
}

```

- Per i dettagli sull'API, consulta la [CompleteMultipartUpload](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **CopyObject** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `CopyObject`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Informazioni di base](#)

Java

SDK per Java 2.x

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Copia un oggetto da un bucket di directory a un altro.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Copies an object from one S3 general purpose bucket to one S3 directory
 * bucket.
 *
 * @param s3Client      The S3 client used to interact with S3
 * @param sourceBucket  The name of the source bucket
 * @param objectKey     The key (name) of the object to be copied
 * @param targetBucket  The name of the target bucket
 */
public static void copyDirectoryBucketObject(S3Client s3Client, String
sourceBucket, String objectKey,
        String targetBucket) {
    logger.info("Copying object: {} from bucket: {} to bucket: {}",
objectKey, sourceBucket, targetBucket);

    try {
        // Create a CopyObjectRequest
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(sourceBucket)
            .sourceKey(objectKey)
            .destinationBucket(targetBucket)
            .destinationKey(objectKey)
            .build();

        // Copy the object
        CopyObjectResponse copyRes = s3Client.copyObject(copyReq);
        logger.info("Successfully copied {} from bucket {} into bucket {}.
CopyObjectResponse: {}",
            objectKey, sourceBucket, targetBucket,
copyRes.copyObjectResult().toString());

    } catch (S3Exception e) {
        logger.error("Failed to copy object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
    }
}
```



```
        throw e;
    }
}
```

- Per i dettagli sull'API, consulta la [CopyObject](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **CreateBucket** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `CreateBucket`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Informazioni di base](#)

Java

SDK per Java 2.x

 Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea un bucket di directory S3.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
```

```
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Creates a new S3 directory bucket in a specified Zone (For example, a
 * specified Availability Zone in this code example).
 *
 * @param s3Client The S3 client used to create the bucket
 * @param bucketName The name of the bucket to be created
 * @param zone The region where the bucket will be created
 * @throws S3Exception if there's an error creating the bucket
 */
public static void createDirectoryBucket(S3Client s3Client, String
bucketName, String zone) throws S3Exception {
    logger.info("Creating bucket: {}", bucketName);

    CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
        .location(LocationInfo.builder()
            .type(LocationType.AVAILABILITY_ZONE)
            .name(zone).build())
        .bucket(BucketInfo.builder()
            .type(BucketType.DIRECTORY)
            .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
            .build())
        .build();

    try {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .createBucketConfiguration(bucketConfiguration).build();
        CreateBucketResponse response = s3Client.createBucket(bucketRequest);
        logger.info("Bucket created successfully with location: {}",
response.location());
    } catch (S3Exception e) {
        logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

```
}  
}
```

- Per i dettagli sull'API, consulta la [CreateBucket](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **CreateMultipartUpload** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `CreateMultipartUpload`.

Java

SDK per Java 2.x

 Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea un caricamento in più parti in un bucket di directory.

```
import com.example.s3.util.S3DirectoryBucketUtils;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;  
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
  
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;  
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;  
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;  
  
/**
```

```

    * This method creates a multipart upload request that generates a unique
    upload
    * ID used to track
    * all the upload parts.
    *
    * @param s3Client    The S3 client used to interact with S3
    * @param bucketName The name of the directory bucket
    * @param objectKey   The key (name) of the object to be uploaded
    * @return The upload ID used to track the multipart upload
    */
    public static String createDirectoryBucketMultipartUpload(S3Client s3Client,
String bucketName, String objectKey) {
        logger.info("Creating multipart upload for object: {} in bucket: {}",
objectKey, bucketName);

        try {
            // Create a CreateMultipartUploadRequest
            CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            // Initiate the multipart upload
            CreateMultipartUploadResponse response =
s3Client.createMultipartUpload(createMultipartUploadRequest);
            String uploadId = response.uploadId();
            logger.info("Multipart upload initiated. Upload ID: {}", uploadId);
            return uploadId;

        } catch (S3Exception e) {
            logger.error("Failed to create multipart upload: {} - Error code:
{}", e.awsErrorDetails().errorMessage(),
                e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
}

```

- Per i dettagli sull'API, consulta la [CreateMultipartUpload](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **CreateSession** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `CreateSession`.

Python

SDK per Python (Boto3)

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

```
class S3ExpressWrapper:
    """Encapsulates Amazon S3 Express One Zone actions using the client
    interface."""

    def __init__(self, s3_client: Any) -> None:
        """
        Initializes the S3ExpressWrapper with an S3 client.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                           access to AWS S3 services.
        """
        self.s3_client = s3_client

    @classmethod
    def from_client(cls) -> "S3ExpressWrapper":
        """
        Creates an S3ExpressWrapper instance with a default s3 client.

        :return: An instance of S3ExpressWrapper initialized with the default S3
client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)
```

```

def create_session(self, bucket_name: str) -> None:
    """
    Creates an express session.
    :param bucket_name: The name of the bucket.
    """
    try:
        self.s3_client.create_session(Bucket=bucket_name)
    except ClientError as client_error:
        logging.error(
            "Couldn't create the express session for bucket %s. Here's why:
%s",
            bucket_name,
            client_error.response["Error"]["Message"],
        )
        raise

```

- Per i dettagli sull'API, consulta [CreateSession AWS SDK for Python \(Boto3\) API Reference](#).

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteBucket** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `DeleteBucket`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Informazioni di base](#)

Java

SDK per Java 2.x

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina un bucket di directory S3.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

/**
 * Deletes the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket to delete
 */
public static void deleteDirectoryBucket(S3Client s3Client, String
bucketName) {
    logger.info("Deleting bucket: {}", bucketName);

    try {
        // Create a DeleteBucketRequest
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Delete the bucket
        s3Client.deleteBucket(deleteBucketRequest);
        logger.info("Successfully deleted bucket: {}", bucketName);
    }
```

```
    } catch (S3Exception e) {  
        logger.error("Failed to delete bucket: {} - Error code: {}",  
            e.awsErrorDetails().errorMessage(),  
            e.awsErrorDetails().errorCode(), e);  
        throw e;  
    }  
}
```

- Per i dettagli sull'API, consulta la [DeleteBucket](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteBucketEncryption** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `DeleteBucketEncryption`.

Java

SDK per Java 2.x

 Note

C'è altro su GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina la configurazione della crittografia per un bucket di directory.

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.DeleteBucketEncryptionRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
```



```
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Deletes the encryption configuration from an S3 bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 */
public static void deleteDirectoryBucketEncryption(S3Client s3Client, String
bucketName) {
    DeleteBucketEncryptionRequest deleteRequest =
DeleteBucketEncryptionRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3Client.deleteBucketEncryption(deleteRequest);
        logger.info("Bucket encryption deleted for bucket: {}", bucketName);
    } catch (S3Exception e) {
        logger.error("Failed to delete bucket encryption: {} - Error code:
{}", e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteBucketEncryption](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteBucketPolicy** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `DeleteBucketPolicy`.

Java

SDK per Java 2.x

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina una policy di bucket per un bucket di directory.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * Deletes the bucket policy for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 */
public static void deleteDirectoryBucketPolicy(S3Client s3Client, String
bucketName) {
    logger.info("Deleting policy for bucket: {}", bucketName);

    try {
        // Create a DeleteBucketPolicyRequest
        DeleteBucketPolicyRequest deletePolicyReq =
DeleteBucketPolicyRequest.builder()
```

```
        .bucket(bucketName)
        .build();

    // Delete the bucket policy
    s3Client.deleteBucketPolicy(deletePolicyReq);
    logger.info("Successfully deleted bucket policy");

    } catch (S3Exception e) {
        logger.error("Failed to delete bucket policy: {} - Error code: {}",
            e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteBucketPolicy](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteObject** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `DeleteObject`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Informazioni di base](#)

Java

SDK per Java 2.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina un oggetto in un bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Deletes an object from the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be deleted
 */
public static void deleteDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
    logger.info("Deleting object: {} from bucket: {}", objectKey,
bucketName);

    try {
        // Create a DeleteObjectRequest
        DeleteObjectRequest deleteObjectRequest =
DeleteObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        // Delete the object
        s3Client.deleteObject(deleteObjectRequest);
    }
```

```
        logger.info("Object {} has been deleted", objectKey);

    } catch (S3Exception e) {
        logger.error("Failed to delete object: {} - Error code: {}",
            e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteObject](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **DeleteObjects** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `DeleteObjects`.

Java

SDK per Java 2.x

 Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elimina più oggetti in un bucket di directory.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectsResponse;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import java.net.URISyntaxException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Deletes multiple objects from the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKeys The list of keys (names) of the objects to be deleted
 */
public static void deleteDirectoryBucketObjects(S3Client s3Client, String
bucketName, List<String> objectKeys) {
    logger.info("Deleting objects from bucket: {}", bucketName);

    try {
        // Create a list of ObjectIdentifier.
        List<ObjectIdentifier> identifiers = objectKeys.stream()
            .map(key -> ObjectIdentifier.builder().key(key).build())
            .toList();

        Delete delete = Delete.builder()
            .objects(identifiers)
            .build();

        DeleteObjectsRequest deleteObjectsRequest =
DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(delete)
            .build();
```

```
        DeleteObjectsResponse deleteObjectsResponse =
s3Client.deleteObjects(deleteObjectsRequest);
        deleteObjectsResponse.deleted().forEach(deleted ->
logger.info("Deleted object: {}", deleted.key()));

    } catch (S3Exception e) {
        logger.error("Failed to delete objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- Per i dettagli sull'API, consulta la [DeleteObjects](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketEncryption** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `GetBucketEncryption`.

Java

SDK per Java 2.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Ottiene la configurazione della crittografia di un bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Retrieves the encryption configuration for an S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return The type of server-side encryption applied to the bucket (e.g.,
 *         AES256, aws:kms)
 */
public static String getDirectoryBucketEncryption(S3Client s3Client, String
bucketName) {
    try {
        // Create a GetBucketEncryptionRequest
        GetBucketEncryptionRequest getRequest =
GetBucketEncryptionRequest.builder()
            .bucket(bucketName)
            .build();

        // Retrieve the bucket encryption configuration
        GetBucketEncryptionResponse response =
s3Client.getBucketEncryption(getRequest);
        ServerSideEncryptionRule rule =
response.serverSideEncryptionConfiguration().rules().get(0);

        String encryptionType =
rule.applyServerSideEncryptionByDefault().sseAlgorithmAsString();
        logger.info("Bucket encryption algorithm: {}", encryptionType);
        logger.info("KMS Customer Managed Key ID: {}",
rule.applyServerSideEncryptionByDefault().kmsMasterKeyID());
        logger.info("Bucket Key Enabled: {}", rule.bucketKeyEnabled());

        return encryptionType;
    } catch (S3Exception e) {
        logger.error("Failed to get bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
    }
}
```



```
        throw e;
    }
}
```

- Per i dettagli sull'API, consulta la [GetBucketEncryption](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetBucketPolicy** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `GetBucketPolicy`.

Java

SDK per Java 2.x

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Ottiene la policy di un bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
```

```
import static
com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

/**
 * Retrieves the bucket policy for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return The bucket policy text
 */
public static String getDirectoryBucketPolicy(S3Client s3Client, String
bucketName) {
    logger.info("Getting policy for bucket: {}", bucketName);

    try {
        // Create a GetBucketPolicyRequest
        GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
            .bucket(bucketName)
            .build();

        // Retrieve the bucket policy
        GetBucketPolicyResponse response =
s3Client.getBucketPolicy(policyReq);

        // Print and return the policy text
        String policyText = response.policy();
        logger.info("Bucket policy: {}", policyText);
        return policyText;

    } catch (S3Exception e) {
        logger.error("Failed to get bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- Per i dettagli sull'API, consulta la [GetBucketPolicy](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetObject** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `GetObject`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nei seguenti esempi di codice:

- [Informazioni di base](#)
- [Creare un URL prefirmato per ottenere un oggetto](#)

Java

SDK per Java 2.x

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Ottiene un oggetto da un bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.charset.StandardCharsets;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
```

```
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Retrieves an object from the specified S3 directory bucket.
 *
 * @param s3Client    The S3 client used to interact with S3
 * @param bucketName  The name of the directory bucket
 * @param objectKey   The key (name) of the object to be retrieved
 * @return            The retrieved object as a ResponseInputStream
 */
public static boolean getDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
    logger.info("Retrieving object: {} from bucket: {}", objectKey,
bucketName);

    try {
        // Create a GetObjectRequest
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(objectKey)
            .bucket(bucketName)
            .build();

        // Retrieve the object as bytes
        ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
        byte[] data = objectBytes.asByteArray();

        // Print object contents to console
        String objectContent = new String(data, StandardCharsets.UTF_8);
        logger.info("Object contents: \n{}", objectContent);

        return true;

    } catch (S3Exception e) {
        logger.error("Failed to retrieve object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        return false;
    }
}
```

```
}
```

- Per i dettagli sull'API, consulta la [GetObject](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **GetObjectAttributes** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `GetObjectAttributes`.

Java

SDK per Java 2.x

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Ottiene gli attributi di un oggetto da un bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesResponse;
import software.amazon.awssdk.services.s3.model.ObjectAttributes;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Retrieves attributes for an object in the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to retrieve attributes for
 * @return True if the object attributes are successfully retrieved, false
 *         otherwise
 */
public static boolean getDirectoryBucketObjectAttributes(S3Client s3Client,
String bucketName, String objectKey) {
    logger.info("Retrieving attributes for object: {} from bucket: {}",
objectKey, bucketName);

    try {
        // Create a GetObjectAttributesRequest
        GetObjectAttributesRequest getObjectAttributesRequest =
GetObjectAttributesRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .objectAttributes(ObjectAttributes.E_TAG,
ObjectAttributes.STORAGE_CLASS,
                ObjectAttributes.OBJECT_SIZE)
            .build();

        // Retrieve the object attributes
        GetObjectAttributesResponse response =
s3Client.getObjectAttributes(getObjectAttributesRequest);
        logger.info("Attributes for object {}:", objectKey);
        logger.info("ETag: {}", response.eTag());
        logger.info("Storage Class: {}", response.storageClass());
        logger.info("Object Size: {}", response.objectSize());
        return true;

    } catch (S3Exception e) {
        logger.error("Failed to retrieve object attributes: {} - Error code:
{}",
            e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
        return false;
    }
}
```

```
}  
}
```

- Per i dettagli sull'API, consulta la [GetObjectAttributes](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **HeadBucket** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `HeadBucket`.

Java

SDK per Java 2.x

 Note

C'è altro su [GitHub](#). Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Verifica se il bucket di directory S3 specificato esiste ed è accessibile.

```
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
  
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;  
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;  
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;  
  
/**  
 * Checks if the specified S3 directory bucket exists and is accessible.
```

```

*
* @param s3Client    The S3 client used to interact with S3
* @param bucketName The name of the directory bucket to check
* @return True if the bucket exists and is accessible, false otherwise
*/
public static boolean headDirectoryBucket(S3Client s3Client, String
bucketName) {
    logger.info("Checking if bucket exists: {}", bucketName);

    try {
        // Create a HeadBucketRequest
        HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();
        // If the bucket doesn't exist, the following statement throws
        NoSuchBucketException,
        // which is a subclass of S3Exception.
        s3Client.headBucket(headBucketRequest);
        logger.info("Amazon S3 directory bucket: \"{}\" found.", bucketName);
        return true;

    } catch (S3Exception e) {
        logger.error("Failed to access bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}

```

- Per i dettagli sull'API, consulta la [HeadBucket](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **HeadObject** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `HeadObject`.

Java

SDK per Java 2.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Ottiene i metadati di un oggetto in un bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Retrieves metadata for an object in the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to retrieve metadata for
 * @return True if the object exists, false otherwise
 */
public static boolean headDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
```

```
logger.info("Retrieving metadata for object: {} from bucket: {}",
objectKey, bucketName);

try {
    // Create a HeadObjectRequest
    HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .build();

    // Retrieve the object metadata
    HeadObjectResponse response = s3Client.headObject(headObjectRequest);
    logger.info("Amazon S3 object: \"{}\" found in bucket: \"{}\" with
ETag: \"{}\"", objectKey, bucketName,
        response.eTag());
    logger.info("Content-Type: {}", response.contentType());
    logger.info("Content-Length: {}", response.contentLength());
    logger.info("Last Modified: {}", response.lastModified());
    return true;

} catch (S3Exception e) {
    logger.error("Failed to retrieve object metadata: {} - Error code:
{}", e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode(), e);
    return false;
}
```

- Per i dettagli sull'API, consulta la [HeadObject](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **ListDirectoryBuckets** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `ListDirectoryBuckets`.

Java

SDK per Java 2.x

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elenca tutti i bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Lists all S3 directory buckets and no general purpose buckets.
 *
 * @param s3Client The S3 client used to interact with S3
 * @return A list of bucket names
 */
public static List<String> listDirectoryBuckets(S3Client s3Client) {
    logger.info("Listing all directory buckets");

    try {
        // Create a ListBucketsRequest
        ListDirectoryBucketsRequest listDirectoryBucketsRequest =
            ListDirectoryBucketsRequest.builder().build();
```

```
// Retrieve the list of buckets
ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listDirectoryBucketsRequest);

// Extract bucket names
List<String> bucketNames = response.buckets().stream()
    .map(Bucket::name)
    .collect(Collectors.toList());

return bucketNames;
} catch (S3Exception e) {
    logger.error("Failed to list buckets: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode());
    throw e;
}
}
```

- Per i dettagli sull'API, consulta la [ListDirectoryBuckets](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **ListMultipartUploads** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `ListMultipartUploads`.

Java

SDK per Java 2.x

Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elenca i caricamenti in più parti in un bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * Lists multipart uploads for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return A list of MultipartUpload objects representing the multipart
 uploads
 */
public static List<MultipartUpload>
listDirectoryBucketMultipartUploads(S3Client s3Client, String bucketName) {
    logger.info("Listing in-progress multipart uploads for bucket: {}",
bucketName);

    try {
        // Create a ListMultipartUploadsRequest
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
```

```
        .bucket(bucketName)
        .build();

    // List the multipart uploads
    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();
    for (MultipartUpload upload : uploads) {
        logger.info("In-progress multipart upload: Upload ID: {}, Key:
{}, Initiated: {}", upload.uploadId(),
            upload.key(), upload.initiated());
    }
    return uploads;

} catch (S3Exception e) {
    logger.error("Failed to list multipart uploads: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode());
    return List.of(); // Return an empty list if an exception is thrown
}
}
```

- Per i dettagli sull'API, consulta la [ListMultipartUploads](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Usa **ListObjectsV2** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `ListObjectsV2`.

Java

SDK per Java 2.x

Note

C'è altro su GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elenca gli oggetti in un bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;

import java.nio.file.Path;
import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Lists objects in the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return A list of object keys in the bucket
 */
```

```
public static List<String> listDirectoryBucketObjectsV2(S3Client s3Client,
String bucketName) {
    logger.info("Listing objects in bucket: {}", bucketName);

    try {
        // Create a ListObjectsV2Request
        ListObjectsV2Request listObjectsV2Request =
ListObjectsV2Request.builder()
                        .bucket(bucketName)
                        .build();

        // Retrieve the list of objects
        ListObjectsV2Response response =
s3Client.listObjectsV2(listObjectsV2Request);

        // Extract and return the object keys
        return response.contents().stream()
                        .map(S3Object::key)
                        .collect(Collectors.toList());

    } catch (S3Exception e) {
        logger.error("Failed to list objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                        e.awsErrorDetails().errorCode());
        throw e;
    }
}
```

- Per i dettagli sull'API, consulta la [ListObjectsversione V2](#) in AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta. [Sviluppo con Amazon S3 utilizzando AWS SDK](#) Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **ListParts** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare **ListParts**.

Java

SDK per Java 2.x

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Elenca le parti di un caricamento in più parti in un bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListPartsRequest;
import software.amazon.awssdk.services.s3.model.ListPartsResponse;
import software.amazon.awssdk.services.s3.model.Part;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * Lists the parts of a multipart upload for the specified S3 directory
 * bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
```

```

    * @param objectKey The key (name) of the object being uploaded
    * @param uploadId The upload ID used to track the multipart upload
    * @return A list of Part representing the parts of the multipart upload
    */
    public static List<Part> listDirectoryBucketMultipartUploadParts(S3Client
s3Client, String bucketName,
        String objectKey, String uploadId) {
        logger.info("Listing parts for object: {} in bucket: {}", objectKey,
bucketName);

        try {
            // Create a ListPartsRequest
            ListPartsRequest listPartsRequest = ListPartsRequest.builder()
                .bucket(bucketName)
                .uploadId(uploadId)
                .key(objectKey)
                .build();

            // List the parts of the multipart upload
            ListPartsResponse response = s3Client.listParts(listPartsRequest);
            List<Part> parts = response.parts();
            for (Part part : parts) {
                logger.info("Uploaded part: Part number = \"{}\", etag = {}",
part.partNumber(), part.eTag());
            }
            return parts;

        } catch (S3Exception e) {
            logger.error("Failed to list parts: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                e.awsErrorDetails().errorCode());
            return List.of(); // Return an empty list if an exception is thrown
        }
    }
}

```

- Per i dettagli sull'API, consulta la [ListParts](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutBucketEncryption** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare **PutBucketEncryption**.

Java

SDK per Java 2.x

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Imposta la crittografia dei bucket per un bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryption;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionByDefault;
import
    software.amazon.awssdk.services.s3.model.ServerSideEncryptionConfiguration;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsClient;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsKey;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.scheduleKeyDeletion;

/**
 * Sets the default encryption configuration for an S3 bucket as SSE-KMS.
 *
 * @param s3Client    The S3 client used to interact with S3
 * @param bucketName  The name of the directory bucket
 * @param kmsKeyId    The ID of the customer-managed KMS key
 */
```

```
public static void putDirectoryBucketEncryption(S3Client s3Client, String
bucketName, String kmsKeyId) {
    // Define the default encryption configuration to use SSE-KMS. For
    directory
    // buckets, AWS managed KMS keys aren't supported. Only customer-managed
    keys
    // are supported.
    ServerSideEncryptionByDefault encryptionByDefault =
    ServerSideEncryptionByDefault.builder()
        .sseAlgorithm(ServerSideEncryption.AWS_KMS)
        .kmsMasterKeyID(kmsKeyId)
        .build();

    // Create a server-side encryption rule to apply the default encryption
    // configuration. For directory buckets, the bucketKeyEnabled field is
    enforced
    // to be true.
    ServerSideEncryptionRule rule = ServerSideEncryptionRule.builder()
        .bucketKeyEnabled(true)
        .applyServerSideEncryptionByDefault(encryptionByDefault)
        .build();

    // Create the server-side encryption configuration for the bucket
    ServerSideEncryptionConfiguration encryptionConfiguration =
    ServerSideEncryptionConfiguration.builder()
        .rules(rule)
        .build();

    // Create the PutBucketEncryption request
    PutBucketEncryptionRequest putRequest =
    PutBucketEncryptionRequest.builder()
        .bucket(bucketName)
        .serverSideEncryptionConfiguration(encryptionConfiguration)
        .build();

    // Set the bucket encryption
    try {
        s3Client.putBucketEncryption(putRequest);
        logger.info("SSE-KMS Bucket encryption configuration set for the
        directory bucket: {}", bucketName);
    } catch (S3Exception e) {
        logger.error("Failed to set bucket encryption: {} - Error code: {}",
        e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode());
    }
}
```

```
        throw e;
    }
}
```

- Per i dettagli sull'API, consulta la [PutBucketEncryption](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutBucketPolicy** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `PutBucketPolicy`.

Java

SDK per Java 2.x

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Applica una policy di bucket a un bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;

/**
```

```

* Sets the following bucket policy for the specified S3 directory bucket.
*<pre>
* {
*     "Version":"2012-10-17",
*     "Statement": [
*         {
*             "Sid": "AdminPolicy",
*             "Effect": "Allow",
*             "Principal": {
*                 "AWS": "arn:aws:iam::<ACCOUNT_ID>:root"
*             },
*             "Action": "s3express:*",
*             "Resource": "arn:aws:s3express:us-west-2:<ACCOUNT_ID>:bucket/
<DIR_BUCKET_NAME>
*         }
*     ]
* }
*</pre>
* This policy grants all S3 directory bucket actions to identities in the
same account as the bucket.
*
* @param s3Client    The S3 client used to interact with S3
* @param bucketName  The name of the directory bucket
* @param policyText  The policy text to be applied
*/
public static void putDirectoryBucketPolicy(S3Client s3Client, String
bucketName, String policyText) {
    logger.info("Setting policy on bucket: {}", bucketName);
    logger.info("Policy: {}", policyText);

    try {
        PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
            .bucket(bucketName)
            .policy(policyText)
            .build();

        s3Client.putBucketPolicy(policyReq);
        logger.info("Bucket policy set successfully!");

    } catch (S3Exception e) {
        logger.error("Failed to set bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}

```

```
}  
}
```

- Per i dettagli sull'API, consulta la [PutBucketPolicy](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **PutObject** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `PutObject`.

Gli esempi di operazioni sono estratti di codice da programmi più grandi e devono essere eseguiti nel contesto. È possibile visualizzare questa operazione nel contesto nel seguente esempio di codice:

- [Informazioni di base](#)

Java

SDK per Java 2.x

 Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Inserisce un oggetto in un bucket di directory.

```
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import software.amazon.awssdk.awscore.exception.AwsErrorDetails;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.PutObjectRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import java.io.UncheckedIOException;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

/**
 * Puts an object into the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be placed in the bucket
 * @param filePath The path of the file to be uploaded
 */
public static void putDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey, Path filePath) {
    logger.info("Putting object: {} into bucket: {}", objectKey, bucketName);

    try {
        // Create a PutObjectRequest
        PutObjectRequest putObj = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        // Upload the object
        s3Client.putObject(putObj, filePath);
        logger.info("Successfully placed {} into bucket {}", objectKey,
bucketName);

    } catch (UncheckedIOException e) {
        throw S3Exception.builder().message("Failed to read the file: " +
e.getMessage()).cause(e)
            .awsErrorDetails(AwsErrorDetails.builder()
                .errorCode("ClientSideException:FailedToReadFile")
                .errorMessage(e.getMessage())
                .build())
            .build();
    } catch (S3Exception e) {
        logger.error("Failed to put object: {}", e.getMessage(), e);
    }
}
```



```
        throw e;
    }
}
```

- Per i dettagli sull'API, consulta la [PutObject](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **UploadPart** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `UploadPart`.

Java

SDK per Java 2.x

 Note

C'è altro su. GitHub Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Carica parti di un caricamento in più parti in un bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.nio.ByteBuffer;
import java.nio.file.Path;
```

```
import java.util.ArrayList;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

/**
 * This method creates part requests and uploads individual parts to S3.
 * While it uses the UploadPart API to upload a single part, it does so
 * sequentially to handle multiple parts of a file, returning all the
 * completed
 * parts.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be uploaded
 * @param uploadId The upload ID used to track the multipart upload
 * @param filePath The path to the file to be uploaded
 * @return A list of uploaded parts
 * @throws IOException if an I/O error occurs
 */
public static List<CompletedPart> multipartUploadForDirectoryBucket(S3Client
s3Client, String bucketName,
    String objectKey, String uploadId, Path filePath) throws IOException
{
    logger.info("Uploading parts for object: {} in bucket: {}", objectKey,
bucketName);

    int partNumber = 1;
    List<CompletedPart> uploadedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

    // Read the local file, break down into chunks and process
    try (RandomAccessFile file = new RandomAccessFile(filePath.toFile(),
"r")) {
        long fileSize = file.length();
```

```
        int position = 0;

        // Sequentially upload parts of the file
        while (position < fileSize) {
            file.seek(position);
            int read = file.getChannel().read(bb);

            bb.flip(); // Swap position and limit before reading from the
buffer
            UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .uploadId(uploadId)
                .partNumber(partNumber)
                .build();

            UploadPartResponse partResponse = s3Client.uploadPart(
                uploadPartRequest,
                RequestBody.fromByteBuffer(bb));

            // Build the uploaded part
            CompletedPart uploadedPart = CompletedPart.builder()
                .partNumber(partNumber)
                .eTag(partResponse.eTag())
                .build();

            // Add the uploaded part to the list
            uploadedParts.add(uploadedPart);

            // Log to indicate the part upload is done
            logger.info("Uploaded part number: {} with ETag: {}", partNumber,
partResponse.eTag());

            bb.clear();
            position += read;
            partNumber++;
        }
    } catch (S3Exception e) {
        logger.error("Failed to list parts: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode());
        throw e;
    }
    return uploadedParts;
}
```

```
}
```

- Per i dettagli sull'API, consulta la [UploadPart](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Utilizzare **UploadPartCopy** con un AWS SDK

Il seguente esempio di codice mostra come utilizzare `UploadPartCopy`.

Java

SDK per Java 2.x

Note

C'è altro su. [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Crea parti di copia in base alla dimensione dell'oggetto di origine e le copia su singole parti in un bucket di directory.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartCopyRequest;
import software.amazon.awssdk.services.s3.model.UploadPartCopyResponse;

import java.io.IOException;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;
```

```

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static
    com.example.s3.util.S3DirectoryBucketUtils.completeDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * Creates copy parts based on source object size and copies over individual
 * parts.
 *
 * @param s3Client      The S3 client used to interact with S3
 * @param sourceBucket  The name of the source bucket
 * @param sourceKey     The key (name) of the source object
 * @param destinationBucket The name of the destination bucket
 * @param destinationKey The key (name) of the destination object
 * @param uploadId      The upload ID used to track the multipart upload
 * @return A list of completed parts
 */
public static List<CompletedPart>
multipartUploadCopyForDirectoryBucket(S3Client s3Client, String sourceBucket,
    String sourceKey, String destinationBucket, String destinationKey,
    String uploadId) {
    // Get the object size to track the end of the copy operation
    HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
        .bucket(sourceBucket)
        .key(sourceKey)
        .build();
    HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
    long objectSize = headObjectResponse.contentLength();

    logger.info("Source Object size: {}", objectSize);

    // Copy the object using 20 MB parts

```

```
        long partSize = 20 * 1024 * 1024; // 20 MB
        long bytePosition = 0;
        int partNum = 1;
        List<CompletedPart> uploadedParts = new ArrayList<>();

        while (bytePosition < objectSize) {
            long lastByte = Math.min(bytePosition + partSize - 1, objectSize -
1);

            logger.info("Part Number: {}, Byte Position: {}, Last Byte: {}",
partNum, bytePosition, lastByte);

            try {
                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                    .sourceBucket(sourceBucket)
                    .sourceKey(sourceKey)
                    .destinationBucket(destinationBucket)
                    .destinationKey(destinationKey)
                    .uploadId(uploadId)
                    .copySourceRange("bytes=" + bytePosition + "-" +
lastByte)

                    .partNumber(partNum)
                    .build();

                UploadPartCopyResponse uploadPartCopyResponse =
s3Client.uploadPartCopy(uploadPartCopyRequest);

                CompletedPart part = CompletedPart.builder()
                    .partNumber(partNum)
                    .eTag(uploadPartCopyResponse.copyPartResult().eTag())
                    .build();
                uploadedParts.add(part);

                bytePosition += partSize;
                partNum++;
            } catch (S3Exception e) {
                logger.error("Failed to copy part number {}: {} - Error code:
{}", partNum,
                    e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode());
                throw e;
            }
        }

        return uploadedParts;
```

```
}
```

- Per i dettagli sull'API, consulta la [UploadPartCopy](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Scenari per l'utilizzo di S3 Directory Bucket AWS SDK

I seguenti esempi di codice mostrano come implementare scenari comuni in S3 Directory Buckets con SDK. AWS Questi scenari illustrano come eseguire attività specifiche chiamando più funzioni all'interno dei bucket di directory S3 o in combinazione con altri Servizi AWS. Ogni scenario include un collegamento al codice sorgente completo, dove è possibile trovare le istruzioni su come configurare ed eseguire il codice.

Gli scenari sono relativi a un livello intermedio di esperienza per aiutarti a comprendere le azioni di servizio nel contesto.

Esempi

- [Crea un URL predefinito per i bucket di directory Amazon S3 per ottenere un oggetto utilizzando un AWS SDK](#)

Crea un URL predefinito per i bucket di directory Amazon S3 per ottenere un oggetto utilizzando un AWS SDK

L'esempio di codice seguente mostra come creare un URL prefirmato per i bucket di directory S3 e ottenere un oggetto.

Java

SDK per Java 2.x

Note

C'è altro su [GitHub](#) Trova l'esempio completo e scopri di più sulla configurazione e l'esecuzione nel [Repository di esempi di codice AWS](#).

Genera un URL GET prefirmato per accedere a un oggetto in un bucket di directory S3.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;

import java.nio.file.Path;
import java.time.Duration;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Presigner;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Generates a presigned URL for accessing an object in the specified S3
 * directory bucket.
 */
```



```
    * @param s3Presigner The S3 presigner client used to generate the presigned
URL
    * @param bucketName The name of the directory bucket
    * @param objectKey The key (name) of the object to access
    * @return A presigned URL for accessing the specified object
    */
    public static String generatePresignedGetURLForDirectoryBucket(S3Presigner
s3Presigner, String bucketName,
        String objectKey) {
        logger.info("Generating presigned URL for object: {} in bucket: {}",
objectKey, bucketName);

        try {
            // Create a GetObjectRequest
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            // Create a GetObjectPresignRequest
            GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(10)) // Presigned URL
valid for 10 minutes
                .getObjectRequest(getObjectRequest)
                .build();

            // Generate the presigned URL
            PresignedGetObjectRequest presignedGetObjectRequest =
s3Presigner.presignGetObject(getObjectPresignRequest);

            // Get the presigned URL
            String presignedURL = presignedGetObjectRequest.url().toString();
            logger.info("Presigned URL: {}", presignedURL);
            return presignedURL;

        } catch (S3Exception e) {
            logger.error("Failed to generate presigned URL: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
}
```

- Per i dettagli sull'API, consulta la [GetObject](#) sezione AWS SDK for Java 2.x API Reference.

Per un elenco completo delle guide per sviluppatori AWS SDK e degli esempi di codice, consulta [Sviluppo con Amazon S3 utilizzando AWS SDK](#). Questo argomento include anche informazioni su come iniziare e dettagli sulle versioni precedenti dell'SDK.

Autenticazione delle richieste (AWS Signature (versione 4))

Argomenti

- [Metodi di autenticazione](#)
- [Introduzione alla firma delle richieste](#)
- [Autenticazione delle richieste: utilizzo dell'intestazione di autorizzazione \(AWS Signature \(versione 4\)\)](#)
- [Autenticazione delle richieste: utilizzo dei parametri di interrogazione \(AWS Signature \(versione 4\)\)](#)
- [Esempi: calcoli delle firme in AWS Signature Version 4](#)
- [Autenticazione delle richieste: Browser-Based caricamenti tramite POST \(AWS Signature \(versione 4\)\)](#)
- [Chiavi specifiche delle policy di autenticazione di Amazon S3 Signature versione 4](#)

Ogni interazione con Amazon S3 è autenticata o anonima. Questa sezione spiega l'autenticazione delle richieste con l'algoritmo AWS Signature Version 4.

Note

Se utilizzi AWS gli SDK (vedi Codice di [esempio e librerie](#)) per inviare le tue richieste, non è necessario leggere questa sezione perché i client SDK autenticano le tue richieste utilizzando le chiavi di accesso fornite. A meno che tu non abbia una buona ragione per non farlo, dovresti sempre usare gli SDK. AWS Nelle regioni che supportano entrambe le versioni di firma, puoi richiedere agli AWS SDK di utilizzare una versione di firma specifica. Per ulteriori informazioni, consulta [Invio di richieste autenticate utilizzando AWS SDK](#). È necessario leggere questa sezione solo se si sta implementando l'algoritmo AWS Signature Version 4 nel client personalizzato.

L'autenticazione con AWS Signature Version 4 fornisce alcuni o tutti i seguenti elementi, a seconda di come scegli di firmare la tua richiesta:

- Verifica dell'identità del richiedente: le richieste autenticate richiedono una firma creata utilizzando le chiavi di accesso (ID della chiave di accesso, chiave di accesso segreta). Per informazioni sull'ottenimento delle chiavi di accesso, consulta [Understanding and Getting Your Security](#)

[Credentials](#) nel. Riferimenti generali di AWS Se stai utilizzando credenziali di sicurezza temporanee, i calcoli della firma richiedono anche un token di sicurezza. Per ulteriori informazioni, consulta [Richiesta di credenziali di sicurezza temporanee](#) nella Guida per l'utente di IAM.

- In-transit protezione dei dati: per evitare di manomettere una richiesta mentre è in transito, si utilizzano alcuni degli elementi della richiesta per calcolare la firma della richiesta. Dopo aver ricevuto la richiesta, Amazon S3 calcola la firma utilizzando gli stessi elementi della richiesta. Se un componente della richiesta ricevuto da Amazon S3 non corrisponde al componente utilizzato per calcolare la firma, Amazon S3 rifiuterà la richiesta.
- Proteggit dal riutilizzo delle parti firmate della richiesta: le parti firmate (utilizzando le AWS firme) delle richieste sono valide entro 15 minuti dal timestamp nella richiesta. Una parte non autorizzata che ha accesso a una richiesta firmata può modificare le parti non firmate della richiesta senza influire sulla validità della richiesta entro 15 minuti. Per questo motivo, ti consigliamo di massimizzare la protezione firmando l'intestazione e il corpo della richiesta, effettuando richieste HTTPS ad Amazon S3 e utilizzando la chiave di `s3:x-amz-content-sha256` condizione (vedi [Chiavi specifiche delle policy di autenticazione di Amazon S3 Signature versione 4](#)) nelle AWS policy per richiedere agli utenti di firmare i corpi delle richieste di Amazon S3.

Note

Amazon S3 supporta complessivamente Signature Version 4, un protocollo per l'autenticazione delle richieste API in entrata ai servizi. AWS Regioni AWS Al momento, Regioni AWS creato prima del 30 gennaio 2014, continuerà a supportare il protocollo precedente, Signature Version 2. Tutte le nuove regioni dopo il 30 gennaio 2014 supporteranno solo la versione 4 di Signature e pertanto tutte le richieste a tali regioni devono essere effettuate con la versione 4 di Signature. Per ulteriori informazioni sulla versione 2 di AWS Signature, consulta la sezione [Signing and Authenticating REST Requests](#) nella Amazon Simple Storage Service User Guide.

Metodi di autenticazione

Puoi esprimere le informazioni di autenticazione utilizzando uno dei seguenti metodi:

- Intestazione di autorizzazione HTTP: l'utilizzo dell'`Authorization` intestazione HTTP è il metodo più comune per autenticare una richiesta Amazon S3. Tutte le operazioni REST di Amazon

S3 (ad eccezione dei caricamenti basati su browser che utilizzano richieste POST) richiedono questa intestazione. Per ulteriori informazioni sul valore dell'Authorization intestazione e su come calcolare la firma e le relative opzioni, consulta [Autenticazione delle richieste: utilizzo dell'intestazione di autorizzazione \(AWS Signature \(versione 4\)\)](#)

- Parametri della stringa di query: puoi utilizzare una stringa di query per esprimere una richiesta interamente in un URL. In questo caso, utilizzi i parametri di interrogazione per fornire le informazioni sulla richiesta, incluse le informazioni di autenticazione. Poiché la firma richiesta è parte dell'URL, questo tipo di URL viene chiamato URL pre-firmato. Puoi utilizzare URL prefirmati per incorporare link cliccabili, validi fino a sette giorni, in HTML. Per ulteriori informazioni, consulta [Autenticazione delle richieste: utilizzo dei parametri di interrogazione \(AWS Signature \(versione 4\)\)](#).

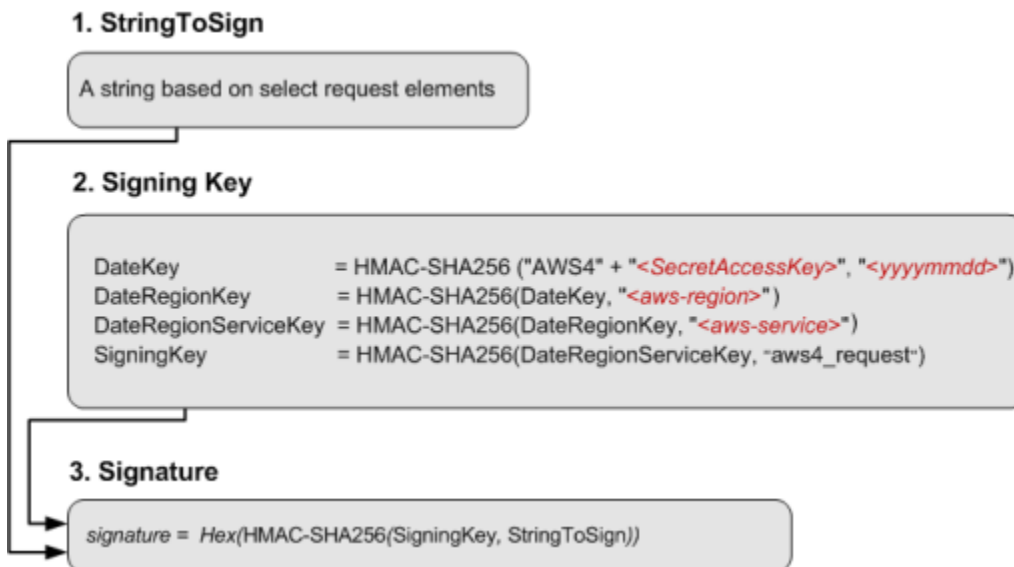
Amazon S3 supporta anche i caricamenti basati su browser che utilizzano richieste HTTP POST. Con una richiesta HTTP POST, puoi caricare contenuti su Amazon S3 direttamente dal browser. Per informazioni sull'autenticazione delle richieste POST, consulta [Browser-Based Caricamenti tramite POST \(AWS Signature \(versione 4\)\)](#)

Introduzione alla firma delle richieste

Le informazioni di autenticazione inviate in una richiesta devono includere una firma. Per calcolare una firma, devi prima concatenare gli elementi della richiesta di selezione per formare una stringa, denominata stringa da firmare. Si utilizza quindi una chiave di firma per calcolare il codice di autenticazione dei messaggi basato su hash (HMAC) della stringa da firmare.

Nella versione 4 di AWS Signature, non si utilizza la chiave di accesso segreta per firmare la richiesta. Invece, si usa prima la propria chiave di accesso segreta per generare una chiave di firma. La chiave di firma derivata è specifica per data, servizio e regione. Per ulteriori informazioni su come derivare una chiave di firma in diversi linguaggi di programmazione, vedi [Esempi di come derivare una chiave di firma per Signature versione 4](#).

Il diagramma seguente illustra il processo generale di calcolo di una firma.



La stringa da firmare dipende dal tipo di richiesta. Ad esempio, quando utilizzi l'intestazione dell'autorizzazione HTTP o i parametri della query per l'autenticazione, utilizzi una combinazione variabile di elementi della richiesta per creare la stringa da firmare. Per una richiesta HTTP POST, la politica POST nella richiesta è la stringa da firmare. Per ulteriori informazioni sul calcolo della stringa da firmare, segui i collegamenti forniti alla fine di questa sezione.

Per chiave di firma, il diagramma mostra una serie di calcoli, dove il risultato di ogni passaggio viene inserito nel passaggio successivo. Il passaggio finale è la chiave di firma.

Dopo aver ricevuto una richiesta autenticata, i server Amazon S3 ricreano la firma utilizzando le informazioni di autenticazione contenute nella richiesta. Se le firme corrispondono, Amazon S3 elabora la richiesta; in caso contrario, la richiesta viene rifiutata.

Per ulteriori informazioni sull'autenticazione delle richieste, consulta i seguenti argomenti:

- [Autenticazione delle richieste: utilizzo dell'intestazione di autorizzazione \(AWS Signature \(versione 4\)\)](#)
- [Autenticazione delle richieste: utilizzo dei parametri di interrogazione \(AWS Signature \(versione 4\)\)](#)
- [Browser-Based Caricamenti tramite POST \(AWS Signature \(versione 4\)\)](#)

Autenticazione delle richieste: utilizzo dell'intestazione di autorizzazione (AWS Signature (versione 4))

Argomenti

- [Panoramica di](#)
- [Calcolo delle firme per l'intestazione di autorizzazione: trasferimento del payload in un unico blocco \(AWS Signature \(versione 4\)\)](#)
- [Calcoli della firma per l'intestazione dell'autorizzazione: trasferimento del payload in più blocchi \(caricamento in blocchi\) \(AWS Signature \(versione 4\)\)](#)
- [Calcoli delle firme per le intestazioni finali \(caricamenti in blocchi\) \(AWS Signature \(versione 4\)\)](#)

Panoramica di

L'utilizzo dell'Authorizationintestazione HTTP è il metodo più comune per fornire informazioni di autenticazione. Ad eccezione delle richieste [POST](#) e delle richieste firmate utilizzando parametri di query, tutte le operazioni di Amazon S3 utilizzano l'intestazione della Authorization richiesta per fornire informazioni di autenticazione.

Di seguito è riportato un esempio del valore per l'intestazione Authorization. Le interruzioni di riga vengono aggiunte a questo esempio solo per migliorare la leggibilità.

```
Authorization: AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,
SignedHeaders=host;range;x-amz-date,
Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024
```

La tabella seguente descrive i vari componenti del valore dell'Authorizationintestazione nell'esempio precedente:

Componente	Description
AWS4-HMAC-SHA256	L'algoritmo utilizzato per calcolare la firma. È necessario fornire questo valore quando si utilizza AWS Signature Version 4 per l'autenticazione. La stringa specifica AWS Signature Version 4 (AWS4) e l'algoritmo di firma (HMAC-SHA256).
Credential	

Componente	Description
	L'ID della chiave di accesso e le informazioni sull'ambito, che includono la data, la regione e il servizio utilizzati per calcolare la firma. Questa stringa ha il seguente formato: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Dove: • <i><date></i> il valore viene specificato utilizzando il YYYYMMDD formato. • <i><aws-service></i> il valore è s3 quando si invia una richiesta ad Amazon S3.
SignedHeaders	Un elenco separato da punto e virgola delle intestazioni di richiesta che hai utilizzato per il calcolo <i>Signature</i> . L'elenco include solo i nomi delle intestazioni e i nomi delle intestazioni devono essere in minuscolo. Ad esempio: <pre>host;range;x-amz-date</pre>

Componente	Description
Signature	La firma a 256 bit espressa come 64 caratteri esadecimali minuscoli. Ad esempio: <pre>fe5f80f77d5fa3beca038a248ff027d0445342fe2855d dc963176630326f1024</pre> Tieni presente che i calcoli della firma variano a seconda dell'opzione scelta per trasferire il payload.

I calcoli delle firme variano a seconda del metodo scelto per trasferire il payload della richiesta. S3 supporta le seguenti opzioni:

- Trasferisci il payload in un unico blocco: in questo caso, sono disponibili le seguenti opzioni di calcolo della firma:
- Opzione payload firmato: puoi facoltativamente calcolare l'intero checksum del payload e includerlo nel calcolo della firma. Ciò fornisce una maggiore sicurezza, ma è necessario leggere il payload due volte o memorizzarlo nel buffer.

Ad esempio, per caricare un file, è necessario prima leggere il file per calcolare un hash del payload per il calcolo della firma e nuovamente per la trasmissione quando si crea la richiesta. Per payload più piccoli, questo approccio potrebbe essere preferibile. Tuttavia, per file di grandi dimensioni, leggere il file due volte può essere inefficiente, quindi potresti voler caricare i dati in blocchi.

Ti consigliamo di includere il checksum del payload per una maggiore sicurezza.

- Opzione payload non firmato: non includere il checksum del payload nel calcolo della firma.

Per istruzioni dettagliate sul calcolo della firma e sulla creazione del valore dell'intestazione di autorizzazione, vedere. [Calcolo delle firme per l'intestazione di autorizzazione: trasferimento del payload in un unico blocco \(AWS Signature \(versione 4\)\)](#)

- Trasferisci il payload in più blocchi (caricamento in blocchi): in questo caso trasferisci il payload in blocchi. Puoi trasferire un payload in blocchi indipendentemente dalle dimensioni del payload.

Puoi suddividere il carico utile in blocchi. Questi possono essere blocchi di dimensioni fisse o variabili. Caricando i dati in blocchi, eviti di leggere l'intero payload per calcolare la firma. Invece, per il primo blocco, si calcola una firma iniziale che utilizza solo le intestazioni della richiesta. Il secondo blocco contiene la firma del primo blocco e ogni blocco successivo contiene la firma del blocco che lo precede. Al termine del caricamento, invii un blocco finale con 0 byte di dati che contiene la firma dell'ultimo blocco del payload. Per ulteriori informazioni, consulta [Calcoli della firma per l'intestazione dell'autorizzazione: trasferimento del payload in più blocchi \(caricamento in blocchi\) \(AWS Signature \(versione 4\)\)](#).

Quando firmi le tue richieste, puoi utilizzare AWS Signature Version 4 o AWS Signature Version 4A. La differenza fondamentale tra le due versioni è determinata dalla modalità di calcolo della firma. Con AWS la versione 4A di Signature, la firma non include Region-specific informazioni e viene calcolata utilizzando l'AWS4-ECDSA-P256-SHA256 algoritmo.

Oltre a queste opzioni, hai la possibilità di includere un rimorchio nella tua richiesta. Per includere un trailer nella tua richiesta, devi specificarlo nell'intestazione impostando `x-amz-content-sha256` il valore appropriato. Se utilizzi un'intestazione finale, devi includere `x-amz-trailer` nell'intestazione e specificare i nomi delle intestazioni finali come stringa in un elenco separato da virgole. Tutte le intestazioni finali vengono scritte dopo il blocco finale. Se stai caricando i dati in più blocchi, devi inviare un blocco finale con 0 byte di dati prima di inviare l'intestazione finale.

Quando invii una richiesta, devi indicare ad Amazon S3 quale delle opzioni precedenti hai scelto nel calcolo della firma, aggiungendo l'intestazione con uno dei seguenti valori: `x-amz-content-sha256`

Valore intestazione	Description
Valore effettivo del checksum del payload	Questo valore è il checksum effettivo dell'oggetto ed è possibile solo quando si caricano i dati in un unico blocco.
UNSIGNED-PAYLOAD	Usalo quando carichi l'oggetto come un singolo blocco non firmato.
STREAMING-UNSIGNED-PAYLOAD-TRAILER	Usalo quando invii un payload non firmato su più blocchi. In questo caso hai anche un'intestazione finale dopo il caricamento del blocco.

Valore intestazione	Description
STREAMING-AWS4-HMAC-SHA256-PAYLOAD	Usalo quando invii un payload su più blocchi e i blocchi vengono firmati utilizzando. AWS4-HMAC-SHA256 Questo produce una firma Sigv4.
STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER	Usalo quando invii un payload su più blocchi e i blocchi vengono firmati utilizzando. AWS4-HMAC-SHA256 Questo produce una firma Sigv4. Inoltre, il digest per i blocchi è incluso come intestazione finale.
STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD	Usalo quando invii un payload su più blocchi e i blocchi vengono firmati utilizzando. AWS4-ECDSA-P256-SHA256 Questo produce una firma SigV4a.
STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD-TRAILER	Usalo quando invii un payload su più blocchi e i blocchi vengono firmati utilizzando. AWS4-ECDSA-P256-SHA256 Questo produce una firma SigV4a. Inoltre, il digest per i blocchi è incluso come intestazione finale.

Dopo aver ricevuto la richiesta, Amazon S3 ricrea la stringa da firmare utilizzando le informazioni nell'intestazione e nell'intestazione. `Authorization date` Quindi verifica con il servizio di autenticazione che le firme corrispondano. La data della richiesta può essere specificata utilizzando `l'HTTP Date` o `l'x-amz-date` intestazione. Se sono presenti entrambe le intestazioni, ha la `x-amz-date` precedenza.

Se le firme corrispondono, Amazon S3 elabora la tua richiesta; in caso contrario, la richiesta non andrà a buon fine.

Per ulteriori informazioni, consulta i seguenti argomenti:

[Calcolo delle firme per l'intestazione di autorizzazione: trasferimento del payload in un unico blocco \(AWS Signature \(versione 4\)\)](#)

[Calcoli della firma per l'intestazione dell'autorizzazione: trasferimento del payload in più blocchi \(caricamento in blocchi\) \(AWS Signature \(versione 4\)\)](#)

[Calcoli delle firme per le intestazioni finali \(caricamenti in blocchi\) \(AWS Signature \(versione 4\)\)](#)

Calcolo delle firme per l'intestazione di autorizzazione: trasferimento del payload in un unico blocco (AWS Signature (versione 4))

Quando si utilizza l'Authorization intestazione per autenticare le richieste, il valore dell'intestazione include, tra le altre cose, una firma. I calcoli della firma variano a seconda della scelta effettuata per il trasferimento del payload (). [Panoramica di](#) Questa sezione spiega i calcoli delle firme quando si sceglie di trasferire il payload in un unico blocco. La sezione di esempio (vedi [Esempi: Calcoli delle firme](#)) mostra i calcoli delle firme e le Authorization intestazioni risultanti che puoi utilizzare come suite di test per verificare il codice.

Important

Quando trasferisci il payload in un unico blocco, puoi opzionalmente scegliere di includere l'hash del payload nei calcoli della firma, denominato payload firmato (se non lo includi, il payload è considerato non firmato). La procedura di firma illustrata nella sezione seguente si applica a entrambi, ma notate le seguenti differenze:

- Opzione di payload firmato: includi l'hash del payload quando crei la richiesta canonica (che poi ne diventa parte integrante StringToSign, come spiegato nella sezione sul calcolo della firma). Inoltre, specificate lo stesso valore dell'`x-amz-content-sha256` intestazione quando inviate la richiesta a S3.
- Opzione di payload non firmato: includi la stringa letterale `UNSIGNED-PAYLOAD` quando crei una richiesta canonica e imposti lo stesso valore dell'`x-amz-content-sha256` intestazione quando invii la richiesta ad Amazon S3.

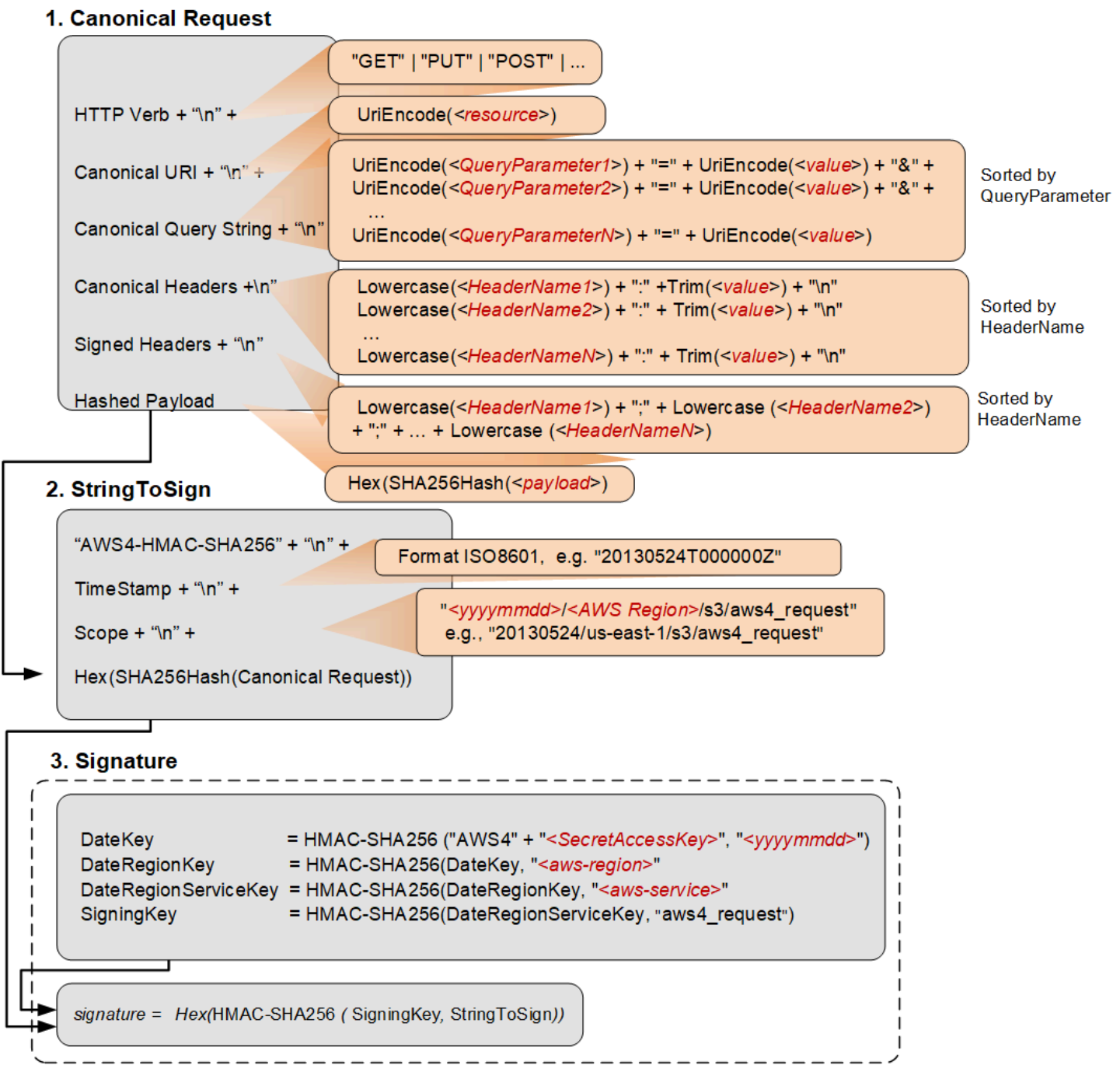
Quando invii la tua richiesta ad Amazon S3, il valore dell'`x-amz-content-sha256` intestazione indica ad Amazon S3 se il payload è firmato o meno. Amazon S3 può quindi creare la firma di conseguenza per la verifica.

In entrambi i casi, poiché il valore dell'`x-amz-content-sha256` intestazione fa già parte dell'intestazione `HashedPayload`, non è necessario includerla come `x-amz-content-sha256` intestazione canonica.

Calcolo di una firma

Per calcolare una firma, occorre una stringa da firmare. Quindi si calcola un HMAC-SHA256 hash della stringa da firmare utilizzando una chiave di firma. Il diagramma seguente illustra il processo, inclusi i vari componenti della stringa creata per la firma

Quando Amazon S3 riceve una richiesta autenticata, calcola la firma e la confronta con la firma fornita nella richiesta. Per questo motivo, è necessario calcolare la firma utilizzando lo stesso metodo utilizzato da Amazon S3. Il processo di invio di una richiesta in un modulo concordato per la firma è chiamato canonicalizzazione.



La tabella seguente descrive le funzioni mostrate nel diagramma. Per queste funzioni devi implementare il codice.

Funzione	Description
Lowercase()	Converte la stringa in minuscolo.

Funzione	Description
Hex()	Codifica in base 16 minuscola.
SHA256Hash()	Funzione hash crittografica Secure Hash Algorithm (SHA).
HMAC-SHA256()	Calcola HMAC utilizzando l'algoritmo SHA256 con la chiave di firma fornita. Questa è la firma definitiva.
Trim()	Rimuove eventuali spazi bianchi all'inizio o alla fine della stringa.

Funzione	Description
<code>UriEncode()</code>	L'URI codifica ogni byte. <code>UriEncode()</code> deve applicare le seguenti regole: • L'URI codifica ogni byte tranne i caratteri senza riserve: 'A'-'Z', 'a'-'z', '0'-'9', '-', '.', '_', e '~'. • Il carattere di spazio è un carattere riservato e deve essere codificato come "%20" (e non come "+"). • Ogni byte codificato in URI è formato da una '%' e dal valore esadecimale a due cifre del byte. • Le lettere nel valore esadecimale devono essere maiuscole, ad esempio "%1A". • Codifica la barra, '/', ovunque tranne nel nome della chiave dell'oggetto. Ad esempio, se il nome della chiave dell'oggetto è <code>photos/Jan/sample.jpg</code>, la barra nel nome della chiave non è codificata.  Important Le <code>UriEncode</code> funzioni standard fornite dalla piattaforma di sviluppo potrebbero non funzionare a causa delle differenze di implementazione e della relativa ambiguità nelle RFC sottostanti. Ti consigliamo di scrivere una <code>UriEncode</code> funzione personalizzata per assicurarti che la codifica funzioni. Di seguito è riportata una funzione <code>example UriEncode ()</code> in Java. <pre>public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBui lder(); for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i);</pre>

Funzione	Description
	<pre> if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); } </pre>

Fase 1. Creazione di una richiesta canonica

Questa sezione fornisce una panoramica sulla creazione di una richiesta canonica.

Di seguito è riportato il formato di richiesta canonico utilizzato da Amazon S3 per calcolare una firma. Affinché le firme corrispondano, devi creare una richiesta canonica in questo formato:

```

<HTTPMethod>\n
<CanonicalURI>\n
<CanonicalQueryString>\n
<CanonicalHeaders>\n
<SignedHeaders>\n
<HashedPayload>

```

Dove:

- **HTTPMethod** è uno dei metodi HTTP, ad esempio GET, PUT, HEAD e DELETE.
- **CanonicalURI** è la URI-encoded versione del componente di percorso assoluto dell'URI: tutto ciò che inizia con «/» che segue il nome di dominio e arriva alla fine della stringa o al carattere del punto interrogativo («?») se disponi di parametri della stringa di interrogazione. L'URI nell'esempio seguente, `/examplebucket/myphoto.jpg`, è il percorso assoluto e non devi codificare «/» nel percorso assoluto:

```
http://s3.amazonaws.com/examplebucket/myphoto.jpg
```

Note

Non si normalizzano i percorsi URI per le richieste ad Amazon S3. Ad esempio, potresti avere un bucket con un oggetto chiamato «my-object//example//photo.user». La normalizzazione del percorso modifica il nome dell'oggetto nella richiesta in «my-object/example/photo.user». Questo è un percorso errato per quell'oggetto.

- **CanonicalQueryString** specifica i parametri della stringa di URI-encoded interrogazione. I URI-encode nomi e i valori vengono assegnati singolarmente. È inoltre necessario ordinare i parametri nella stringa di query canonica in ordine alfabetico in base al nome della chiave. L'ordinamento avviene dopo la codifica. La stringa di query nel seguente esempio di URI è `prefix=somePrefix&marker=someMarker&max-keys=20`:

```
http://s3.amazonaws.com/examplebucket?prefix=somePrefix&marker=someMarker&max-keys=20
```

La stringa di query canonica è la seguente (le interruzioni di riga vengono aggiunte a questo esempio a fini di leggibilità):

```
UriEncode("marker")+"="+UriEncode("someMarker")+"&" +  
UriEncode("max-keys")+"="+UriEncode("20") + "&" +  
UriEncode("prefix")+"="+UriEncode("somePrefix")
```

Quando una richiesta ha come target una sottorisorsa, il valore del parametro di query corrispondente sarà una stringa vuota (""). Ad esempio, il seguente URI identifica la sottorisorsa ACL sul bucket `examplebucket`:

```
http://s3.amazonaws.com/examplebucket?acl
```

CanonicalQueryString In questo caso è la seguente:

```
UriEncode("acl") + "=" + ""
```

Se l'URI non include un '?', la richiesta non contiene una stringa di query e occorre impostare la stringa di query canonica su una stringa vuota (""). Dovrai comunque includere "\n".

- *CanonicalHeaders* è un elenco di intestazioni delle richieste con i relativi valori. Le singole coppie di nome e valore dell'intestazione sono separate dal carattere di nuova riga ("\n"). I nomi delle intestazioni devono essere in minuscolo. È necessario ordinare alfabeticamente i nomi delle intestazioni per creare la stringa, come mostrato nell'esempio seguente:

```
Lowercase(<HeaderName1>)+":"+Trim(<value>)+"\n"
Lowercase(<HeaderName2>)+":"+Trim(<value>)+"\n"
...
Lowercase(<HeaderNameN>)+":"+Trim(<value>)+"\n"
```

Le funzioni Lowercase() e Trim() utilizzate in questo esempio sono descritte nella sezione precedente.

L'*CanonicalHeaders* elenco deve includere quanto segue:

- Intestazione host HTTP.
- Se l'Content-MD5 intestazione è presente nella richiesta, è necessario aggiungerla all'*CanonicalHeaders* elenco.
- Devi aggiungere anche qualsiasi intestazione x-amz-* che desideri includere nella richiesta. Ad esempio, se utilizzi credenziali di sicurezza temporanee, nella tua richiesta devi includere x-amz-security-token. È necessario aggiungere questa intestazione nell'elenco di *CanonicalHeaders*.

Note

- L'x-amz-content-sha256 intestazione è obbligatoria per tutte le richieste di AWS Signature Version 4. Fornisce un hash del payload di richiesta. Tuttavia, non è necessario includerla x-amz-content-sha256 come intestazione canonica perché S3 utilizzerà automaticamente il suo valore nel calcolo dell'hash del payload inviato nella richiesta.
- Se non è presente alcun payload, devi indicare l'hash di una stringa vuota.

- Se non vuoi che S3 controlli l'hash della richiesta, puoi invece utilizzare la stringa letterale. "UNSIGNED-PAYLOAD"

Le seguenti sono stringhe di esempio. CanonicalHeaders I nomi di intestazione sono in caratteri minuscoli e in ordine alfabetico.

Example Esempio 1

```
host:s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130708T220855Z
```

Example Esempio 2

```
host:s3.amazonaws.com
x-amz-content-sha256:UNSIGNED-PAYLOAD
x-amz-date:20130708T220855Z
```

Example Esempio 3

```
host:s3.amazonaws.com
x-amz-date:20130708T220855Z
```

Note

Ai fini del calcolo di una firma di autorizzazione, sono obbligatori l'intestazione dell'host e tutte le x-amz-* intestazioni, escluse x-amz-content-sha256. Tuttavia, per prevenire la manomissione dei dati, dovresti considerare di includere tutte le intestazioni nel calcolo della firma.

La firma dell'x-amz-content-sha256 intestazione è facoltativa perché S3 utilizzerà il suo valore per calcolare l'hash del payload della richiesta ricevuta.

- **SignedHeaders** è un elenco in ordine alfabetico e separato da punto e virgola di nomi di intestazione delle richieste in lettere minuscole. Le intestazioni della richiesta nell'elenco sono le stesse che hai incluso nella stringa CanonicalHeaders. Negli esempi precedenti, il valore di sarebbe il seguente: **SignedHeaders**

Example Esempi 1 e 2

```
host;x-amz-content-sha256;x-amz-date
```

Example Esempio 3

```
host;x-amz-date
```

- **HashedPayload** è il valore esadecimale dell'hash SHA256 del payload della richiesta.

```
Hex(SHA256Hash(<payload>))
```

Se non è presente alcun payload nella richiesta, calcola un hash della stringa vuota come segue:

```
Hex(SHA256Hash(""))
```

L'hash restituisce i seguenti valori:

```
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Ad esempio, quando carichi un oggetto utilizzando una richiesta PUT, fornisci i dati dell'oggetto nel corpo. Quando si recupera un oggetto utilizzando una richiesta GET, si calcola l'hash della stringa vuota.

Fase 2: creazione di una stringa di firma

Questa sezione fornisce una panoramica sulla creazione di una stringa da firmare. Per istruzioni dettagliate, vedi [Attività 2: Creare una stringa per accedere](#) al. Riferimenti generali di AWS

La stringa da firmare è una concatenazione delle seguenti stringhe:

```
"AWS4-HMAC-SHA256" + "\n" +  
timestampISO8601Format + "\n" +  
<Scope> + "\n" +  
Hex(SHA256Hash(<CanonicalRequest>))
```

La stringa costante AWS4-HMAC-SHA256 specifica l'algoritmo hash che stai utilizzando,. HMAC-SHA256 timeStampÈ l'ora UTC corrente nel formato ISO 8601 (ad esempio,). 20130524T000000Z

Scopelega la firma risultante a una data, una data e un servizio Regione AWS specifici. Pertanto, la firma risultante funzionerà solo nella regione specifica e per un servizio specifico. La firma è valida per sette giorni dopo la data specificata.

```
date.Format(<YYYYMMDD>) + "/" + <region> + "/" + <service> + "/aws4_request"
```

Per Amazon S3, la stringa di servizio ès3. Per un elenco di *region* stringhe, consulta [Regioni ed endpoint](#) in. Riferimenti generali di AWS La colonna Regione in questa tabella fornisce l'elenco delle stringhe Region valide.

L'ambito seguente limita la firma risultante alla us-east-1 regione e ad Amazon S3.

```
20130606/us-east-1/s3/aws4_request
```

Note

Scopedeve utilizzare la stessa data utilizzata per calcolare la chiave di firma, come illustrato nella sezione seguente.

Attività 3: Calcolare la firma

Nella versione 4 di AWS Signature, invece di utilizzare le chiavi di AWS accesso per firmare una richiesta, è necessario creare innanzitutto una chiave di firma con ambito per una regione e un servizio specifici. Per ulteriori informazioni sulla firma delle chiavi, consulta[Introduzione alla firma delle richieste](#).

```
DateKey           = HMAC-SHA256("AWS4"+"<SecretAccessKey>", "<YYYYMMDD>")
DateRegionKey     = HMAC-SHA256(<DateKey>, "<aws-region>")
DateRegionServiceKey = HMAC-SHA256(<DateRegionKey>, "<aws-service>")
SigningKey        = HMAC-SHA256(<DateRegionServiceKey>, "aws4_request")
```

Note

Alcuni casi d'uso possono elaborare le chiavi di firma per un massimo di 7 giorni. Per ulteriori informazioni, consulta [Condividere un oggetto con altri](#).

Per un elenco delle stringhe di regione, vedi [Regioni ed endpoint](#) in. Riferimenti generali di AWS

L'utilizzo di una chiave di firma consente di conservare le AWS credenziali in un unico posto sicuro. Ad esempio, se hai più server che comunicano con Amazon S3, condividi la chiave di firma con quei server; non devi conservare una copia della tua chiave di accesso segreta su ogni server. La chiave di firma è valida per un massimo di sette giorni. Quindi, ogni volta che calcoli la chiave di firma, dovrai condividerla con i tuoi server. Per ulteriori informazioni, consulta [Autenticazione delle richieste \(AWS Signature \(versione 4\)\)](#).

La firma finale è l' HMAC-SHA256 hash della stringa da firmare, utilizzando la chiave di firma come chiave.

```
HMAC-SHA256(SigningKey, StringToSign)
```

Per istruzioni dettagliate sulla creazione di una firma, vedi [Attività 3: Creare una firma in](#). Riferimenti generali di AWS

Esempi: Calcoli delle firme

È possibile utilizzare gli esempi in questa sezione come riferimento per controllare i calcoli delle firme nel codice. I calcoli mostrati negli esempi utilizzano i seguenti dati:

- Esempi di chiavi di accesso.

Parametro	Valore
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

- Richiedi il timestamp di 20130524T000000Z (). Fri, 24 May 2013 00:00:00 GMT
- Nome `examplebucket` del secchio.
- Si presume che il bucket si trovi nella regione degli Stati Uniti orientali (Virginia settentrionale). La credenziale Scope e i Signing Key calcoli vengono utilizzati `us-east-1` come identificatore della regione. Per informazioni su altre regioni, vedere [Regioni ed endpoint](#) nel Riferimenti generali di AWS.
- È possibile utilizzare richieste in stile path o in stile host virtuale. Gli esempi seguenti mostrano come firmare una richiesta in stile host virtuale, ad esempio:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

Per ulteriori informazioni, consulta [Virtual Hosting of Buckets](#) nella Amazon Simple Storage Service User Guide.

Esempio: GET Object

L'esempio seguente ottiene i primi 10 byte di un oggetto (`test.txt`) da `examplebucket`. Per ulteriori informazioni sull'azione API, consulta [GetObject](#).

```
GET /test.txt HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
Range: bytes=0-9
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date: 20130524T000000Z
```

Poiché questa richiesta GET non fornisce alcun contenuto, il `x-amz-content-sha256` valore può essere l'hash del corpo della richiesta vuoto o la stringa letterale. "UNSIGNED-PAYLOAD" I passaggi seguenti mostrano i calcoli delle firme e la costruzione dell'Authorization intestazione utilizzando l'hash di una stringa vuota.

1. StringToSign

a. CanonicalRequest

```
GET
```



```
/test.txt
```

```
host:examplebucket.s3.amazonaws.com
range:bytes=0-9
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z
```

```
host;range;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Nella stringa di richiesta canonica, l'ultima riga è l'hash del corpo vuoto della richiesta. La terza riga è vuota perché non ci sono parametri di interrogazione nella richiesta.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
7344ae5b7ee6c3e7e6b0fe0640412a37625d1fbfff95c48bbb2dc43964946972
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Firma

```
f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41
```

4. Intestazione di autorizzazione

L'Authorizationintestazione risultante è la seguente:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;range;x-amz-content-sha256;x-amz-
date,Signature=f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41
```

Esempio: oggetto PUT

Questo esempio di richiesta PUT crea un oggetto (test\$file.text) inexamplebucket.

L'esempio presuppone quanto segue:

- Stai richiedendo REDUCED_REDUNDANCY come classe di archiviazione aggiungendo l'intestazione della x-amz-storage-class richiesta. Per informazioni sulle classi di storage, consulta [Storage Classes](#) nella Amazon Simple Storage Service User Guide.
- Il contenuto del file caricato è una stringa. "Welcome to Amazon S3." Il valore di x-amz-content-sha256 nella richiesta si basa su questa stringa.

Per informazioni sull'azione API, consulta [PutObject](#).

```
PUT test$file.text HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Fri, 24 May 2013 00:00:00 GMT
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-storage-class: REDUCED_REDUNDANCY
x-amz-content-sha256: 44ce7dd67c959e0d3524ffac1771dfbba87d2b6b4b4e99e42034a8b803f8b072

<Payload>
```

I passaggi seguenti mostrano i calcoli delle firme.

1. StringToSign

a. CanonicalRequest

```
PUT
/test%24file.text

date:Fri, 24 May 2013 00:00:00 GMT
host:examplebucket.s3.amazonaws.com
x-amz-content-sha256:44ce7dd67c959e0d3524ffac1771dfbba87d2b6b4b4e99e42034a8b803f8b072
```

```
x-amz-date:20130524T000000Z
x-amz-storage-class:REDUCED_REDUNDANCY

date;host;x-amz-content-sha256;x-amz-date;x-amz-storage-class
44ce7dd67c959e0d3524ffac1771dfbba87d2b6b4b4e99e42034a8b803f8b072
```

Nella richiesta canonica, la terza riga è vuota perché nella richiesta non sono presenti parametri di interrogazione. L'intestazione `x-amz-content-sha256` canonica può essere firmata facoltativamente poiché l'hash del payload è già fornito nella parte inferiore della richiesta. L'ultima riga è l'hash del corpo, che deve essere uguale al `x-amz-content-sha256` header valore inviato a S3 nella richiesta HTTP.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
9e0e90d9c76de8fa5b200d8c849cd5b8dc7a3be3951ddb7f6a76b4158342019d
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Firma

```
98ad721746da40c64f1a55b78f14c238d841ea1380cd77a1b5971af0ece108bd
```

4. Intestazione di autorizzazione

L'`Authorization` intestazione risultante è la seguente:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/
aws4_request,SignedHeaders=date;host;x-amz-content-sha256;x-amz-date;x-amz-storage-
class,Signature=98ad721746da40c64f1a55b78f14c238d841ea1380cd77a1b5971af0ece108bd
```

Esempio: GET Bucket Lifecycle

La seguente richiesta GET recupera la configurazione del ciclo di vita di `examplebucket`. Per informazioni sull'azione API, consulta [GetBucketLifecycleConfiguration](#).

```
GET ?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Poiché la richiesta non fornisce alcun contenuto del corpo, il valore dell'`x-amz-content-sha256` intestazione è l'hash del corpo della richiesta vuoto. I passaggi seguenti mostrano i calcoli delle firme.

1. StringToSign

a. CanonicalRequest

```
GET
/
lifecycle=
host:examplebucket.s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Nella richiesta canonica, l'ultima riga è l'hash del corpo della richiesta vuoto.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
9766c798316ff2757b517bc739a67f6213b4ab36dd5da2f94eaebf79c77395ca
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
  "<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Firma

```
fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

4. Intestazione di autorizzazione

L'Authorization intestazione risultante è la seguente:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-
date,Signature=fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

Esempio: Get Bucket (List Objects)

L'esempio seguente recupera un elenco di oggetti dal `examplebucket` bucket. Per informazioni sull'azione API, consulta [ListObjects](#)

```
GET ?max-keys=2&prefix=J HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Poiché la richiesta non fornisce un corpo, il valore di `x-amz-content-sha256` è l'hash del corpo della richiesta vuoto. I passaggi seguenti mostrano i calcoli delle firme.

1. StringToSign

a. CanonicalRequest

```
GET
/
max-keys=2&prefix=J
host:examplebucket.s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

Nella stringa canonica, l'ultima riga è l'hash del corpo vuoto della richiesta.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
df57d21db20da04d7fa30298dd4488ba3a2b47ca3a489c74750e0f1e7df1b9b7
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Firma

```
34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```

4. Intestazione di autorizzazione

L'Authorizationintestazione risultante è la seguente:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-
date,Signature=34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```


Calcoli della firma per l'intestazione dell'autorizzazione: trasferimento del payload in più blocchi (caricamento in blocchi) (AWS Signature (versione 4))

Come descritto in [Panoramica di](#), quando si autenticano le richieste utilizzando l'Authorizationintestazione, è possibile caricare il payload in blocchi. È possibile inviare dati in blocchi di dimensioni fisse o variabili. Questa sezione descrive il processo di calcolo della firma nel caricamento in blocchi, come si crea il corpo del blocco e come funziona la firma ritardata nel momento in cui si carica per la prima volta il blocco e si invia la firma nel blocco successivo. La sezione di esempio (vedi [Esempio: oggetto PUT](#)) mostra i calcoli delle firme e le Authorizationintestazioni risultanti che puoi utilizzare come suite di test per verificare il codice.

Note

Quando si trasferiscono dati in una serie di blocchi, è necessario effettuare una delle seguenti operazioni:

- Specificate in modo esplicito la lunghezza totale del contenuto (lunghezza dell'oggetto in byte più metadati in ogni blocco) utilizzando l'intestazione HTTP. Content-Length A tale scopo, è necessario precalcolare la lunghezza totale del payload, inclusi i metadati inviati in ogni blocco, prima di iniziare la richiesta.
- Specificate l'intestazione HTTP. Transfer-Encoding Se includi l'Transfer-Encodingintestazione e specifichi un valore diverso daidentity, devi omettere l'intestazione. Content-Length

Per tutte le richieste, è necessario includere l'x-amz-decoded-content-lengthintestazione, specificando la dimensione dell'oggetto in byte.

Ogni calcolo della firma del blocco include la firma del blocco precedente. Per iniziare, crei una firma iniziale utilizzando solo le intestazioni. La firma iniziale viene utilizzata nel calcolo della firma del primo blocco. Per ogni blocco successivo, si crea una firma in blocco che include la firma del blocco precedente. Pertanto, le firme dei blocchi vengono concatenate; ovvero, la firma del blocco n è una funzione $F(\text{blocco } n, \text{firma}(\text{blocco } n-1))$. Il concatenamento garantisce l'invio dei blocchi nell'ordine corretto.

Per eseguire un caricamento in blocchi, procedi come segue:

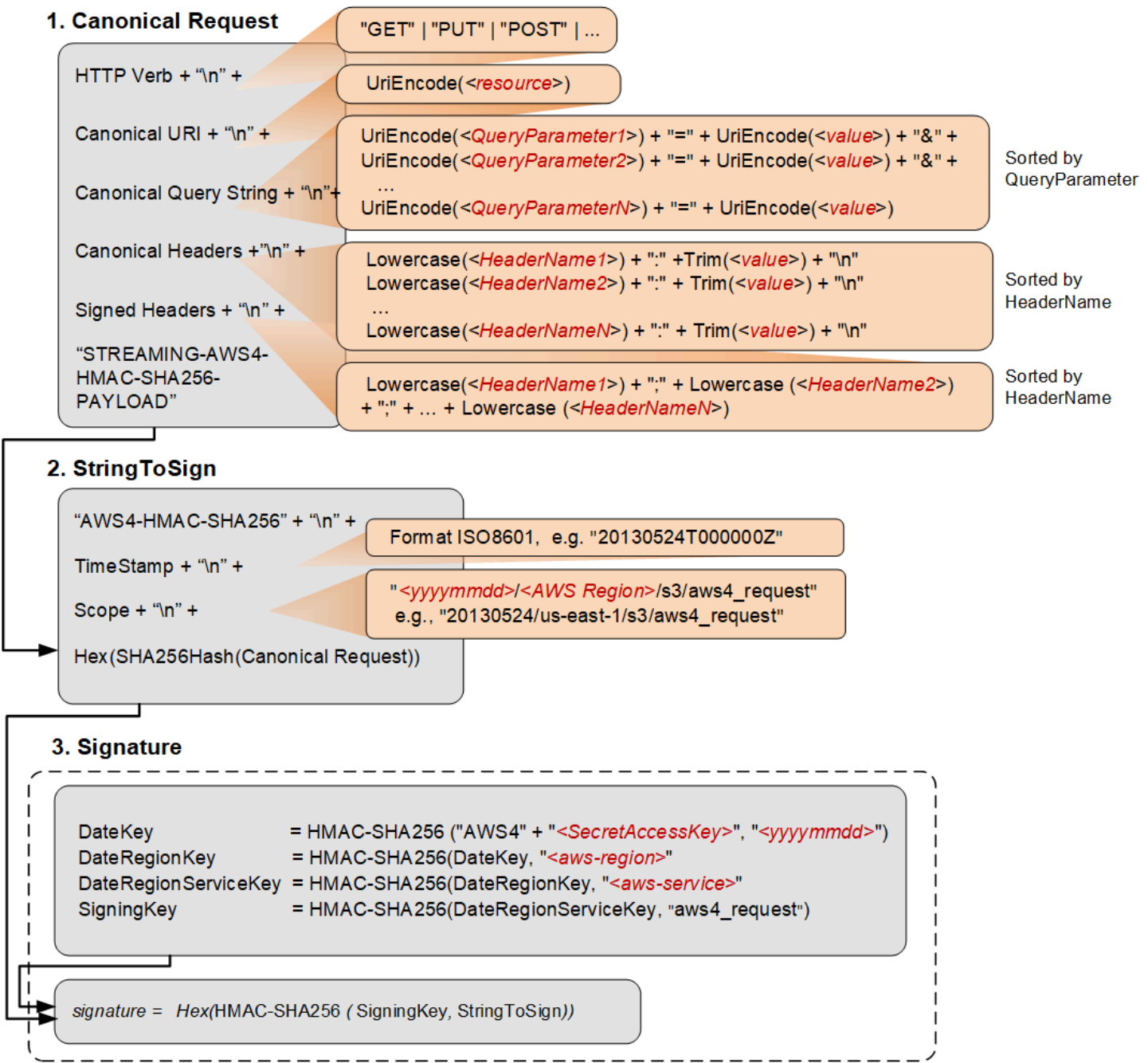
1. Decidi la dimensione del blocco del payload. Ne hai bisogno quando scrivi il codice.

La dimensione del blocco deve essere di almeno 8 KB. Consigliamo una dimensione del blocco di almeno 64 KB per prestazioni migliori. Questa dimensione del blocco si applica a tutti i blocchi tranne l'ultimo. L'ultimo blocco inviato può essere inferiore a 8 KB. Se il payload è piccolo e può essere contenuto in un blocco, può essere inferiore agli 8 KB.

2. Crea la firma iniziale da includere nel primo blocco. Per ulteriori informazioni, consulta [Calcolo della firma iniziale](#).
3. Crea il primo blocco e trasmettilo in streaming. Per ulteriori informazioni, consulta [Definizione del corpo del blocco](#).
4. Per ogni blocco successivo, calcola la firma del blocco che include la firma precedente nella stringa che firmi, costruisci il blocco e invialo. Per ulteriori informazioni, consulta [Definizione del corpo del blocco](#).
5. Inviare l'ultimo blocco aggiuntivo, che è lo stesso degli altri blocchi nella costruzione, ma contiene zero byte di dati. Per ulteriori informazioni, consulta [Definizione del corpo del blocco](#).

Calcolo della firma iniziale

Il diagramma seguente illustra il processo di calcolo della firma iniziale.



La tabella seguente descrive le funzioni mostrate nel diagramma. Per queste funzioni devi implementare il codice.

Funzione	Description
Lowercase()	Converte la stringa in minuscolo.
Hex()	Codifica in base 16 minuscola.

Funzione	Description
SHA256Hash()	Funzione hash crittografica Secure Hash Algorithm (SHA).
HMAC-SHA256()	Calcola HMAC utilizzando l'algoritmo SHA256 con la chiave di firma fornita. Questa è la firma definitiva.
Trim()	Rimuove eventuali spazi bianchi all'inizio o alla fine della stringa.

Funzione	Description
<code>UriEncode()</code>	L'URI codifica ogni byte. <code>UriEncode()</code> deve applicare le seguenti regole: • L'URI codifica ogni byte tranne i caratteri senza riserve: 'A'-'Z', 'a'-'z', '0'-'9', '-', '.', '_', e '~'. • Il carattere di spazio è un carattere riservato e deve essere codificato come "%20" (e non come "+"). • Ogni byte codificato in URI è formato da una '%' e dal valore esadecimale a due cifre del byte. • Le lettere nel valore esadecimale devono essere maiuscole, ad esempio "%1A". • Codifica la barra, '/', ovunque tranne nel nome della chiave dell'oggetto. Ad esempio, se il nome della chiave dell'oggetto è <code>photos/Jan/sample.jpg</code>, la barra nel nome della chiave non è codificata.  Important Le <code>UriEncode</code> funzioni standard fornite dalla piattaforma di sviluppo potrebbero non funzionare a causa delle differenze di implementazione e della relativa ambiguità nelle RFC sottostanti. Ti consigliamo di scrivere una <code>UriEncode</code> funzione personalizzata per assicurarti che la codifica funzioni. Di seguito è riportata una funzione <code>example UriEncode ()</code> in Java. <pre>public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBui lder(); for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i);</pre>

Funzione	Description
	<pre> if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); } </pre>

Per informazioni sul processo di firma, vedere [Calcolo delle firme per l'intestazione di autorizzazione: trasferimento del payload in un unico blocco \(AWS Signature \(versione 4\)\)](#). Il processo è lo stesso, tranne per il fatto che la creazione di `CanonicalRequest` è diversa come segue:

- Oltre alle intestazioni delle richieste che intendi aggiungere, devi includere le seguenti intestazioni:

Header	Description
x-amz-content-sha256	Questa intestazione è obbligatoria per tutte le richieste di AWS Signature Version 4. Imposta il valore su <code>STREAMING-AWS4-HMAC-SHA256-PAYLOAD</code> per indicare che la firma copre solo le intestazioni e che non è presente alcun payload.

Header	Description
Content-Encoding	Impostare il valore su <code>aws-chunked</code> . Amazon S3 supporta più valori di codifica dei contenuti. Puoi specificare la codifica dei contenuti personalizzata quando usi l'API di streaming Signature Version 4. Ad esempio: Content-Encoding : aws-chunked,gzip Amazon S3 memorizza l'oggetto risultante senza il <code>aws-chunked</code> valore nell'intestazione. <code>content-encoding</code> Se <code>aws-chunked</code> è l'unico valore che passi nell'<code>content-encoding</code> intestazione, S3 considera l'intestazione vuota e non restituisce questa <code>content-encoding</code> intestazione quando recuperi l'oggetto.
x-amz-decoded-content-length	Imposta il valore sulla lunghezza, in byte, dei dati da suddividere in blocchi, senza contare i metadati. Ad esempio, se stai caricando un file da 4 GB, imposta il valore su 4294967296. Questa è la dimensione grezza dell'oggetto da caricare (dati che desideri archiviare in Amazon S3).
Content-Length	Imposta il valore sulla dimensione effettiva del corpo HTTP trasmesso , che include la lunghezza dei dati (valore impostato per <code>x-amz-decoded-content-length</code>), più i metadati in blocchi. Ogni blocco contiene metadati, come la firma del blocco precedente. I calcoli dei blocchi sono descritti nella sezione seguente. Se si include l'<code>Transfer-Encoding</code> intestazione e si specifica un valore diverso da <code>identity</code>, non è necessario includere l'<code>Content-Length</code> intestazione.

Invii il primo blocco con la firma iniziale. È necessario costruire il blocco come descritto nella sezione seguente.

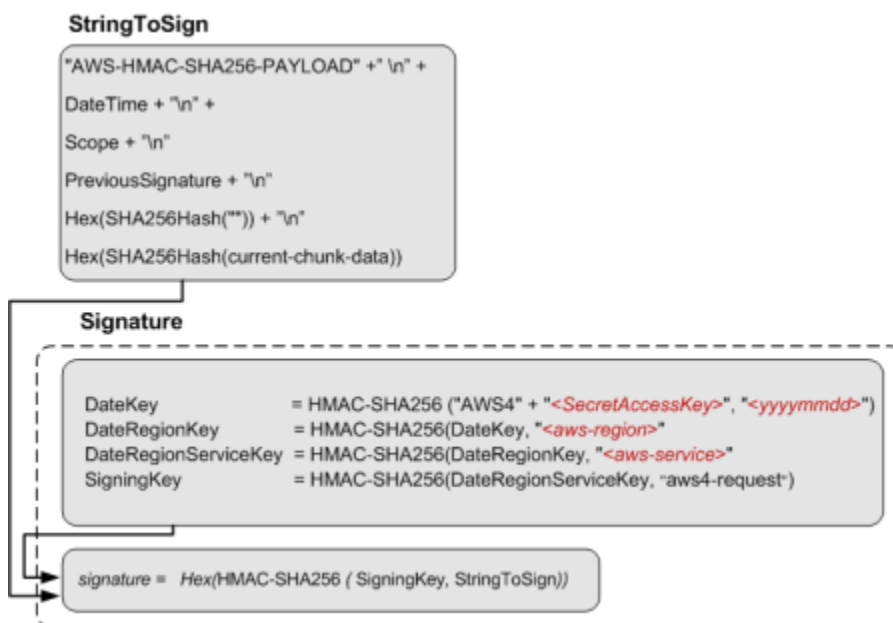
Definizione del corpo del blocco

Tutti i blocchi includono alcuni metadati. Ogni blocco deve essere conforme alla seguente struttura:

```
string(IntHexBase(chunk-size)) + ";chunk-signature=" + signature + \r\n + chunk-data + \r\n
```

Dove:

- `IntHexBase()` è una funzione che si scrive per convertire la dimensione di un blocco intero in formato esadecimale. Ad esempio, se la dimensione del blocco è 65536, la stringa esadecimale è «10000».
- *chunk-size* è la dimensione, in byte, del blocco di dati, senza metadati. Ad esempio, se si carica un oggetto da 65 KB e si utilizza una dimensione del blocco di 64 KB, si caricano i dati in tre blocchi: il primo sarà 64 KB, il secondo 1 KB e l'ultimo blocco con 0 byte.
- *signature* Per ogni blocco, si calcola la firma utilizzando la seguente stringa da firmare. Per il primo blocco, si utilizza la firma iniziale come firma precedente.



La dimensione dei dati del blocco finale che invii è 0, sebbene il corpo del blocco contenga ancora metadati, inclusa la firma del blocco precedente.

Esempio: oggetto PUT

È possibile utilizzare gli esempi in questa sezione come riferimento per controllare i calcoli delle firme nel codice. Prima di esaminare gli esempi, tenete presente quanto segue:

- I calcoli delle firme in questi esempi utilizzano le seguenti credenziali di sicurezza di esempio.

Parametro	Valore
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

- Tutti gli esempi utilizzano il timestamp della richiesta 20130524T000000Z (). Fri, 24 May 2013 00:00:00 GMT
- Tutti `examplebucket` gli esempi utilizzano come nome del bucket.
- Si presume che il bucket si trovi nella regione Stati Uniti orientali (Virginia settentrionale) e le credenziali Scope e i Signing Key calcoli vengono utilizzati `us-east-1` come identificatore della regione. Per ulteriori informazioni consultare l'articolo relativo alle [Regioni ed endpoint](#) nel documento Riferimenti generali di Amazon Web Services.
- Puoi utilizzare lo stile del percorso o le richieste di stile ospitate virtualmente. Gli esempi seguenti utilizzano richieste di stile ospitate virtualmente, ad esempio:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

Per ulteriori informazioni, consulta [Virtual Hosting of Buckets](#) nella Amazon Simple Storage Service User Guide.

L'esempio seguente invia una PUT richiesta per caricare un oggetto. I calcoli delle firme presuppongono quanto segue:

- Stai caricando un file di testo da 65 KB e il contenuto del file è una stringa di un carattere composta dalla lettera «a».

- La dimensione del blocco è 64 KB. Di conseguenza, il payload viene caricato in tre blocchi, 64 KB, 1 KB e l'ultimo blocco con 0 byte di dati in blocco.
- L'oggetto risultante ha il nome della chiave. `chunkObject.txt`
- La richiesta viene effettuata `REDUCED_REDUNDANCY` come classe di archiviazione aggiungendo l'intestazione della `x-amz-storage-class` richiesta.

Per informazioni sull'azione API, consulta [PutObject](#). La sintassi generale della richiesta è la seguente:

```
PUT /examplebucket/chunkObject.txt HTTP/1.1
Host: s3.amazonaws.com
x-amz-date: 20130524T000000Z
x-amz-storage-class: REDUCED_REDUNDANCY
Authorization: SignatureToBeCalculated
x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-PAYLOAD
Content-Encoding: aws-chunked
x-amz-decoded-content-length: 66560
Content-Length: 66824
<Payload>
```

I passaggi seguenti mostrano i calcoli delle firme.

1. Firma iniziale: crea una stringa da firmare

a. CanonicalRequest

```
PUT
/examplebucket/chunkObject.txt

content-encoding:aws-chunked
content-length:66824
host:s3.amazonaws.com
x-amz-content-sha256:STREAMING-AWS4-HMAC-SHA256-PAYLOAD
x-amz-date:20130524T000000Z
x-amz-decoded-content-length:66560
x-amz-storage-class:REDUCED_REDUNDANCY

content-encoding;content-length;host;x-amz-content-sha256;x-amz-date;x-amz-decoded-content-length;x-amz-storage-class
```

```
STREAMING-AWS4-HMAC-SHA256-PAYLOAD
```

Nella richiesta canonica, la terza riga è vuota perché nella richiesta non sono presenti parametri di interrogazione. L'ultima riga è la stringa costante fornita come valore del Payload con hash, che deve essere uguale al valore di `x-amz-content-sha256` header

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
cee3fed04b70f867d036f722359b0b1f2f0e5dc0efadbc082b76c4c60e316455
```

Note

Per informazioni su ciascuna riga della stringa da firmare, consulta il diagramma che spiega il calcolo della firma iniziale.

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Firma del seme

```
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
```

4. Intestazione di autorizzazione

L'intestazione di autorizzazione risultante è la seguente:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/
aws4_request,SignedHeaders=content-encoding;content-length;host;x-amz-
```

```
content-sha256;x-amz-date;x-amz-decoded-content-length;x-amz-storage-
class,Signature=4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
```

5. Blocco 1: (65536 byte, con valore 97 per la lettera «a»)

a. Stringa in blocchi da firmare:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

 Note

Per informazioni su ogni riga della stringa da firmare, vedete il diagramma precedente che mostra i vari componenti della stringa da firmare (ad esempio, le ultime tre righe sono, `previous-signaturehash("")`, `ehash(current-chunk-data)`).

b. Firma in blocchi:

```
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
```

c. Dati in blocco inviati:

```
10000;chunk-
signature=ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
<65536-bytes>
```

6. Chunk 2: (1024 byte, con valore 97 per la lettera 'a')

a. Stringa in blocchi da firmare:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

```
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. Firma in blocco:

```
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497
```

c. Dati in blocco inviati:

```
400; chunk-  
signature=0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497  
<1024 bytes>
```

7. Blocco 3: (dati a 0 byte)

a. Stringa in blocchi da firmare:

```
AWS4-HMAC-SHA256-PAYLOAD  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497  
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855  
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

b. Firma in blocco:

```
b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

c. Dati in blocco inviati:

```
0; chunk-  
signature=b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

Calcoli delle firme per le intestazioni finali (caricamenti in blocchi) (AWS Signature (versione 4))

Quando autentichi le richieste utilizzando l'Authorizationintestazione, puoi anche caricare il payload in blocchi. Quando invii i dati per l'oggetto in blocchi, hai anche la possibilità di includere le intestazioni finali. Per ulteriori informazioni, consulta [Calcoli della firma per l'intestazione dell'autorizzazione: trasferimento del payload in più blocchi \(caricamento in blocchi\) \(AWS Signature \(versione 4\)\)](#). Questa sezione descrive i passaggi da eseguire per includere un'intestazione finale alla fine del caricamento di più blocchi.

Important

Quando includi le intestazioni finali, devi inviare quanto segue nell'intestazione iniziale:

- È necessario `x-amz-content-sha256` impostare un valore appropriato che indichi che verrà incluso un trailer. Per vedere i valori accettabili per `x-amz-content-sha256`, consulta [Autenticazione delle richieste: utilizzo dell'intestazione di autorizzazione \(AWS Signature \(versione 4\)\)](#).
- Devi `x-amz-trailer` impostarlo in modo da indicare i contenuti che includi nell'intestazione finale.

Le intestazioni finali vengono inviate solo dopo che i blocchi sono stati caricati. I blocchi precedenti vengono inviati normalmente e firmati come descritto nelle sezioni precedenti, incluso l'invio del blocco finale con un payload di 0 byte. Le intestazioni finali vengono incluse come blocco a sé stante e inviate dopo il blocco finale con un payload di 0 byte. Ad esempio, se i tuoi dati terminano con un blocco di 100 KB, invierai quanto segue:

- Blocchi di dati precedenti
- Parte finale dell'oggetto da 100 KB
- Blocco da 0 byte che indica la fine dell'oggetto
- Blocco di intestazioni finali

Esempi: controllo dei calcoli delle firme

È possibile utilizzare gli esempi in questa sezione come riferimento per controllare i calcoli delle firme nel codice. Prima di esaminare gli esempi, tenete presente quanto segue:

- I calcoli delle firme in questi esempi utilizzano le seguenti credenziali di sicurezza di esempio.

Parametro	Valore
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

- Tutti gli esempi utilizzano il timestamp della richiesta 20130524T000000Z (). Fri, 24 May 2013 00:00:00 GMT
- Tutti `examplebucket` gli esempi utilizzano come nome del bucket.
- Si presume che il bucket si trovi nella regione Stati Uniti orientali (Virginia settentrionale) e le credenziali Scope e i Signing Key calcoli vengono utilizzati `us-east-1` come identificatore della regione. Per ulteriori informazioni, consulta la pagina relativa a [regioni ed endpoint](#) nei Riferimenti generali di Amazon Web Services.
- Puoi utilizzare lo stile del percorso o le richieste di stile ospitate virtualmente. Gli esempi seguenti utilizzano richieste di stile ospitate virtualmente, ad esempio:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

Per ulteriori informazioni, consulta [Virtual Hosting of Buckets](#) nella Amazon Simple Storage Service User Guide.

L'esempio seguente invia una PUT richiesta per caricare un oggetto. I calcoli delle firme presuppongono quanto segue:

- Stai caricando un file di testo da 65 KB e il contenuto del file è una stringa di un carattere composta dalla lettera «a».
- La dimensione del blocco è 64 KB. Di conseguenza, il payload viene caricato in tre blocchi, 64 KB, 1 KB e l'ultimo blocco con 0 byte di dati in blocco.

- L'oggetto risultante ha il nome della chiave. `chunkObject.txt`
- La richiesta viene effettuata `REDUCED_REDUNDANCY` come classe di archiviazione aggiungendo l'intestazione della `x-amz-storage-class` richiesta.
- Il trasferimento include un valore di checksum CRC32C come intestazione finale.

Per informazioni sull'azione API, consulta. [PutObject](#) La sintassi generale della richiesta è la seguente:

```
PUT /examplebucket/chunkObject.txt HTTP/1.1
Host: s3.amazonaws.com
x-amz-date: 20130524T000000Z
x-amz-storage-class: REDUCED_REDUNDANCY
Authorization: SignatureToBeCalculated
x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER
Content-Encoding: aws-chunked
x-amz-decoded-content-length: 66560
x-amz-trailer: x-amz-checksum-crc32c
Content-Length: 66946
<Payload>
```

I passaggi seguenti mostrano i calcoli delle firme.

1. Firma iniziale: crea una stringa da firmare

a. CanonicalRequest

```
PUT
/examplebucket/chunkObject.txt

content-encoding:aws-chunked
host:s3.amazonaws.com
x-amz-content-sha256:STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER
x-amz-date:20130524T000000Z
x-amz-decoded-content-length:66560
x-amz-storage-class:REDUCED_REDUNDANCY
x-amz-trailer:x-amz-checksum-crc32c

content-encoding;host;x-amz-content-sha256;x-amz-date;x-amz-decoded-content-length;x-amz-storage-class;x-amz-trailer
```

```
STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER
```

Nella richiesta canonica, la terza riga è vuota perché nella richiesta non sono presenti parametri di interrogazione. L'ultima riga è la stringa costante fornita come valore del Payload con hash, che deve essere uguale al valore di `x-amz-content-sha256` header

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
44d48b8c2f70eae815a0198cc73d7a546a73a93359c070abbaa5e6c7de112559
```

Note

Per informazioni su ciascuna riga della stringa da firmare, consulta il diagramma che spiega il calcolo della firma iniziale.

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Firma del seme

```
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
```

4. Intestazione di autorizzazione

L'intestazione di autorizzazione risultante è la seguente:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/
aws4_request,SignedHeaders=content-encoding;content-length;host;x-amz-
```



```
content-sha256;x-amz-date;x-amz-decoded-content-length;x-amz-storage-
class,Signature=106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
```

5. Blocco 1: (65536 byte, con valore 97 per la lettera «a»)

a. Stringa in blocchi da firmare:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

Note

Per informazioni su ogni riga della stringa da firmare, vedete il diagramma nell'argomento precedente ([Calcolo della firma iniziale](#)) che mostra i vari componenti della stringa da firmare. Ad esempio, le ultime tre righe sono costituite da quanto segue:

- *previous-signature*
- `hash("")`
- `hash(current-chunk-data)`

b. Firma in blocco:

```
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
```

c. Dati in blocco inviati:

```
10000;chunk-
signature=b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
<65536-bytes>
```

6. Blocco 2: (1024 byte, con valore 97 per la lettera «a»)

a. Stringa in blocchi da firmare:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. Firma in blocco:

```
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
```

c. Dati in blocco inviati:

```
400;chunk-
signature=1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
<1024-bytes>
```

7. Blocco 3: (dati a 0 byte)

a. Stringa in blocchi da firmare:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

b. Firma in blocco:

```
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

c. Dati in blocco inviati:

```
0;chunk-
signature=2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

8. Parte 4: intestazioni finali

a. Stringa del trailer da firmare:

```

AWS4-HMAC-SHA256-TRAILER
20130524T000000Z
20130524/us-east-1/s3/aws4_request
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
1e376db7e1a34a8ef1c4bcee131a2d60a1cb62503747488624e10995f448d774

```

Note

Le ultime due righe sono `previous-signature` (la firma del blocco di dati da 0 byte), hash (*`trailing-checksum-header-name: base64-encoded-trailing-checksum-value`*`\n`).

L'hash viene calcolato come segue senza spazi bianchi:

- Il nome dell'intestazione del checksum finale
- I due punti () :
- Il valore di checksum finale codificato in base64
- Un carattere di nuova riga (). `\n`

In questo esempio, in cui si utilizza l'algoritmo `crc32c` hash, con il valore di checksum codificato in base64 `s008/Q==`, è possibile rappresentare il calcolo come segue: `hash( 'x-amz-checksum-crc32c:s008/Q==\n' )`

b. Firma in blocchi:

```
d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435
```

c. Dati in blocco inviati:

```

x-amz-checksum-crc32c:s008/Q==
x-amz-trailer-
signature:d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435

```

Autenticazione delle richieste: utilizzo dei parametri di interrogazione (AWS Signature (versione 4))

Come descritto nella panoramica sull'autenticazione (vedere [Metodi di autenticazione](#)), è possibile fornire informazioni di autenticazione utilizzando i parametri della stringa di query. L'utilizzo dei parametri di interrogazione per autenticare le richieste è utile quando si desidera esprimere una richiesta interamente in un URL. Questo metodo viene anche chiamato prefirma di un URL.

Uno scenario d'uso per gli URL prefirmati è la possibilità di concedere un accesso temporaneo alle risorse Amazon S3. Ad esempio, puoi incorporare un URL prefirmato nel tuo sito Web o, in alternativa, utilizzarlo in un client a riga di comando (come Curl) per scaricare oggetti.

Note

Puoi anche usare il AWS CLI per creare URL prefirmati. Per ulteriori informazioni, consulta [presign](#) nella documentazione di riferimento dei comandi della AWS CLI .

Di seguito è riportato un esempio di URL prefirmato.

```
https://examplebucket.s3.amazonaws.com/test.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=<your-access-key-id>/20130721/us-east-1/s3/aws4_request
&X-Amz-Date=20130721T201207Z
&X-Amz-Expires=86400
&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

Nell'URL di esempio, nota quanto segue:

- I feed di riga vengono aggiunti per motivi di leggibilità.
- Il valore X-Amz-Credential nell'URL mostra il carattere “/” solo per leggibilità. In pratica, dovrebbe essere codificato come. %2F Ad esempio:

```
&X-Amz-Credential=<your-access-key-id>%2F20130721%2Fus-east-1%2Fs3%2Faws4_request
```

La tabella seguente descrive i parametri della query nell'URL che forniscono informazioni di autenticazione.

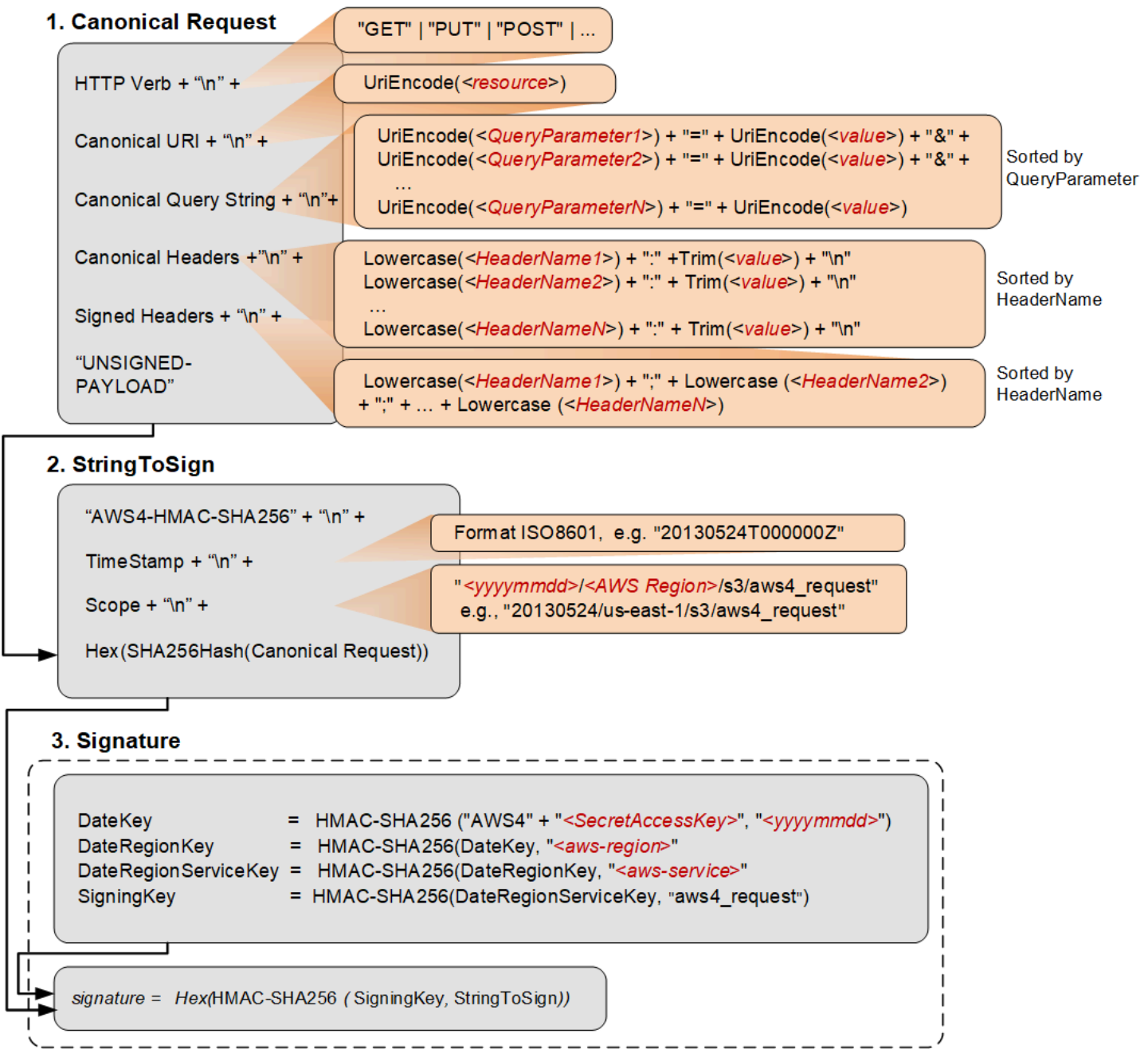
Nome parametro stringa di query	Valore di esempio
X-Amz-Algorithm	Identifica la versione di AWS Signature e l'algoritmo utilizzato per calcolare la firma. Per la versione 4 di AWS Signature, imposti il valore di questo parametro su <code>AWS4-HMAC-SHA256</code>. Questa stringa identifica a AWS la versione 4 (AWS 4) di Signature e l' HMAC-SHA256 algoritmo (HMAC-SHA256).
X-Amz-Credential	Oltre all'ID della chiave di accesso, questo parametro fornisce anche l'ambito (Regione AWS e il servizio) per cui la firma è valida. Questo valore deve corrispondere all'ambito utilizzato nei calcoli della firma, illustrato nella sezione seguente. La forma generale per questo valore di parametro è la seguente: <pre><your-access-key-id> /<date>/<Regione AWS>/<AWS-service> /aws4_request</pre> Ad esempio: <pre>AKIAIOSFODNN7EXAMPLE/20130721/us-east-1/s3/aws4_request</pre> Per Amazon S3, la <code>AWS-service</code> stringa è <code>s3</code>. Per un elenco delle <code>AWS-region</code> stringhe S3, consulta Regioni ed endpoint in. Riferimenti generali di AWS
X-Amz-Date	Il formato di data e ora deve seguire lo standard ISO 8601 e deve essere formattato con il formato <code>"yyyyMMddT Z". HHmmss</code>

Nome parametro stringa di query	Valore di esempio
	Ad esempio, se la data e l'ora erano "08/01/2016 15:32:41.982-700", devono prima essere convertite in UTC (Coordinated Universal Time) e quindi inviate come «20160801T223241Z».
X-Amz-Expires	Fornisce il periodo di tempo, in secondi, per il quale l'URL prefirmato generato è valido. Ad 86400 esempio, (24 ore). Questo valore è un numero intero. Il valore minimo che puoi impostare è 1 e il massimo è 604800 (sette giorni). Un URL prefirmato può essere valido per un massimo di sette giorni perché la chiave di firma utilizzata nel calcolo della firma è valida per un massimo di sette giorni.
X-Amz-SignedHeaders	Elenca le intestazioni che hai utilizzato per calcolare la firma. Per i calcoli delle firme sono necessarie le seguenti intestazioni: • L'hostintestazione HTTP. • Tutte le x-amz-* intestazioni che intendi aggiungere alla richiesta.  Note Per maggiore sicurezza, devi firmare tutte le intestazioni della richiesta che intendi includere nella richiesta.

Nome parametro stringa di query	Valore di esempio
X-Amz-Signature	Fornisce la firma per autenticare la richiesta. Questa firma deve corrispondere alla firma calcolata da Amazon S3; in caso contrario, Amazon S3 respinge la richiesta. Ad esempio, 733255ef022bec3f2a8701cd61d4b371f3f28c9f193a1f02279211d48d5193d7 I calcoli delle firme sono descritti nella sezione seguente.
X-Amz-Security-Token	Parametro opzionale delle credenziali se si utilizzano credenziali provenienti dal servizio STS.

Calcolo di una firma

Il diagramma seguente illustra il processo di calcolo della firma.



La tabella seguente descrive le funzioni mostrate nel diagramma. Per queste funzioni devi implementare il codice.

Funzione	Description
Lowercase()	Converte la stringa in minuscolo.
Hex()	Codifica in base 16 minuscola.

Funzione	Description
SHA256Hash()	Funzione hash crittografica Secure Hash Algorithm (SHA).
HMAC-SHA256()	Calcola HMAC utilizzando l'algoritmo SHA256 con la chiave di firma fornita. Questa è la firma definitiva.
Trim()	Rimuove eventuali spazi bianchi all'inizio o alla fine della stringa.

Funzione	Description
<code>UriEncode()</code>	L'URI codifica ogni byte. <code>UriEncode()</code> deve applicare le seguenti regole: • L'URI codifica ogni byte tranne i caratteri senza riserve: 'A'-'Z', 'a'-'z', '0'-'9', '-', '.', '_', e '~'. • Il carattere di spazio è un carattere riservato e deve essere codificato come "%20" (e non come "+"). • Ogni byte codificato in URI è formato da una '%' e dal valore esadecimale a due cifre del byte. • Le lettere nel valore esadecimale devono essere maiuscole, ad esempio "%1A". • Codifica la barra, '/', ovunque tranne nel nome della chiave dell'oggetto. Ad esempio, se il nome della chiave dell'oggetto è <code>photos/Jan/sample.jpg</code>, la barra nel nome della chiave non è codificata.  Important Le <code>UriEncode</code> funzioni standard fornite dalla piattaforma di sviluppo potrebbero non funzionare a causa delle differenze di implementazione e della relativa ambiguità nelle RFC sottostanti. Ti consigliamo di scrivere una <code>UriEncode</code> funzione personalizzata per assicurarti che la codifica funzioni. Di seguito è riportata una funzione <code>example UriEncode ()</code> in Java. <pre>public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBui lder(); for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i);</pre>

Funzione	Description
	<pre> if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); } </pre>

Per ulteriori informazioni sul processo di firma (dettagli sulla creazione di una richiesta canonica, stringa da firmare e calcoli della firma), vedere. [Calcolo delle firme per l'intestazione di autorizzazione: trasferimento del payload in un unico blocco \(AWS Signature \(versione 4\)\)](#) Il processo è generalmente lo stesso, tranne per il fatto che la creazione di CanonicalRequest un URL prefirmato è diversa come segue:

- Non includi un hash del payload nella richiesta canonica, perché quando crei un URL prefirmato, non conosci il contenuto del payload perché l'URL viene utilizzato per caricare un payload arbitrario. Invece, usi una stringa costante. UNSIGNED-PAYLOAD
- La Canonical Query String deve includere tutti i parametri di query della tabella precedente ad eccezione di X-Amz-Signature
- Per S3, è necessario includere il parametro di X-Amz-Security-Token query nell'URL se si utilizzano credenziali provenienti dal servizio STS.
- Le intestazioni canoniche devono includere l'intestazione HTTP. host Se prevedi di includere una qualsiasi delle intestazioni, queste x-amz-* intestazioni devono essere aggiunte anche per il calcolo della firma. Facoltativamente, puoi aggiungere tutte le altre intestazioni che intendi includere nella richiesta. Per una maggiore sicurezza, dovresti firmare quante più intestazioni possibile. Se aggiungi un'intestazione firmata che è anche un parametro di query con firma e il valore differisce, riceverai un InvalidRequest errore poiché l'input è in conflitto.

Un esempio

Supponiamo di avere un oggetto `test.txt` nel `examplebucket` secchio. Vuoi condividere questo oggetto con altri per un periodo di 24 ore (86400 secondi) creando un URL prefirmato.

```
https://examplebucket.s3.amazonaws.com/test.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request
&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

I passaggi seguenti illustrano prima i calcoli della firma e poi la costruzione dell'URL prefirmato. L'esempio formula i seguenti presupposti aggiuntivi:

- Il timestamp della richiesta è. `Fri, 24 May 2013 00:00:00 GMT`
- Il bucket si trova nella regione Stati Uniti orientali (Virginia settentrionale) e le credenziali Scope e i Signing Key calcoli vengono utilizzati `us-east-1` come identificatori della regione. Per ulteriori informazioni, consulta la sezione relativa a [regioni ed endpoint](#) nella Riferimenti generali di AWS.

È possibile utilizzare questo esempio come test case per verificare la firma calcolata dal codice; tuttavia, è necessario utilizzare lo stesso nome del bucket, chiave oggetto, timestamp e le seguenti credenziali di esempio:

Parametro	Valore
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

1. StringToSign

a. CanonicalRequest

```
GET
```

```

/test.txt
X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE
%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
host:examplebucket.s3.amazonaws.com

host
UNSIGNED-PAYLOAD

```

b. StringToSign

```

AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
3bfa292879f6447bbcd47001decf97f4a54dc650c8942174ae0a9121cf58ad04

```

2. SigningKey

```

signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")

```

3. Firma

```

aeeed9bbccd4d02ee5c0109b86d86835f995330da4c265957d157751f604d404

```

Ora hai tutte le informazioni per creare un URL prefirmato. L'URL risultante per questo esempio è mostrato come segue (puoi usarlo per confrontare il tuo URL prefirmato):

```

https://examplebucket.s3.amazonaws.com/test.txt?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=aeeed9bbccd4d02ee5c0109b86d86835f995330da4c265957d157751f604d404

```


Authorization per l'autenticazione, vedere. [Autenticazione delle richieste: utilizzo dell'intestazione di autorizzazione \(AWS Signature \(versione 4\)\)](#)

- Oggetto GET: questo esempio genera un URL prefirmato per ottenere un oggetto. I parametri di interrogazione forniscono la firma e altre informazioni di autenticazione. Gli utenti possono incollare un URL prefirmato nel browser per recuperare l'oggetto oppure utilizzare l'URL per creare un link cliccabile. Per informazioni sull'utilizzo dei parametri di interrogazione per l'autenticazione, vedere. [Autenticazione delle richieste: utilizzo dei parametri di interrogazione \(AWS Signature \(versione 4\)\)](#)

Il resto di questa sezione descrive gli esempi in Java e C#. Gli argomenti includono istruzioni per scaricare gli esempi e per eseguirli.

Esempi di calcolo delle firme utilizzando Java (AWS Signature (versione 4))

L'esempio Java che mostra il calcolo della firma può essere scaricato all'indirizzo <https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AWSS3SigV4JavaSamples.zip>. Nota che questi esempi utilizzano AWS SDK for Java v1 e consigliamo di utilizzare [AWS SDK for Java v2 per le nuove applicazioni](#). In `RunAllSamples.java`, la `main()` funzione esegue richieste di esempio per creare un oggetto, recuperare un oggetto e creare un URL prefirmato per l'oggetto. L'esempio crea un oggetto dalla stringa di testo fornita nel codice:

```
PutS3ObjectSample.putS3Object(bucketName, regionName, awsAccessKey, awsSecretKey);
GetS3ObjectSample.getS3Object(bucketName, regionName, awsAccessKey, awsSecretKey);
PresignedUrlSample.getPresignedUrlToS3Object(bucketName, regionName, awsAccessKey,
awsSecretKey);
PutS3ObjectChunkedSample.putS3ObjectChunked(bucketName, regionName, awsAccessKey,
awsSecretKey);
```

Per testare gli esempi su un Linux-based computer

Le seguenti istruzioni si riferiscono al sistema operativo Linux.

1. In un terminale, accedi alla directory che contiene `AWSS3SigV4JavaSamples.zip`.
2. Decomprimere il file `.zip`.
3. In un editor di testo, apri il file `./com/amazonaws/services/s3/samples/RunAllSamples.java`. Aggiorna il codice con le seguenti informazioni:

- Il nome di un bucket in cui è possibile creare il nuovo oggetto.

 Note

Gli esempi utilizzano una richiesta di stile ospitata virtualmente per accedere al bucket. Per evitare potenziali errori, assicurati che il nome del bucket sia conforme alle regole di denominazione dei bucket, come spiegato in [Restrizioni e limitazioni del bucket](#) nella Amazon Simple Storage Service User Guide.

- Regione AWS dove si trova il bucket.

Se il bucket si trova nella regione degli Stati Uniti orientali (Virginia settentrionale), usa `us-east-1` per specificare la regione. Per un elenco di altre Regioni AWS, vai [su Amazon Simple Storage Service \(S3\) nel](#). Riferimenti generali di AWS

4. Compila il codice sorgente e memorizza le classi compilate nella directory `bin/`

```
javac -d bin -source 6 -verbose com
```

5. Cambia la directory in `bin/`, quindi esegui `RunAllSamples`

```
java com.amazonaws.services.s3.sample.RunAllSamples
```

Il codice esegue tutti i metodi in `main()`. Per ogni richiesta, l'output mostrerà la richiesta canonica, la stringa da firmare e la firma.

Esempi di calcoli delle firme utilizzando C# (AWS Signature (versione 4))

L'esempio in C# che mostra il calcolo della firma può essere scaricato all'indirizzo. https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AmazonS3SigV4_Samples_CSharp.zip Nota che questi esempi utilizzano AWS SDK per .NET v1 e consigliamo di utilizzare [AWS SDK per .NET v3](#) per le nuove applicazioni. In `Program.cs`, la `main()` funzione esegue richieste di esempio per creare un oggetto, recuperare un oggetto e creare un URL prefirmato per l'oggetto. Il codice per il calcolo della firma si trova nella cartella. `\Signers`

```
PutS3ObjectSample.Run(awsRegion, bucketName, "MySampleFile.txt");

Console.WriteLine("\n\n*****");
PutS3ObjectChunkedSample.Run(awsRegion, bucketName, "MySampleFileChunked.txt");
```



```
Console.WriteLine("\n\n*****");
GetS3ObjectSample.Run(awsRegion, bucketName, "MySampleFile.txt");

Console.WriteLine("\n\n*****");
PresignedUrlSample.Run(awsRegion, bucketName, "MySampleFile.txt");
```

Per testare gli esempi con Microsoft Visual Studio 2010 o versioni successive

1. Decomprimere il file .zip.
2. Avvia Visual Studio, quindi apri il file .sln.
3. Aggiorna il App.config file con credenziali di sicurezza valide.
4. Aggiorna il codice come segue:
 - InProgram.cs, fornisci il nome del bucket e il Regione AWS luogo in cui risiede il bucket. L'esempio crea un oggetto in questo bucket.
5. Esegui il codice.
6. Per verificare che l'oggetto sia stato creato, copia l'URL prefirmato creato dal programma e incollalo in una finestra del browser.

Autenticazione delle richieste: Browser-Based caricamenti tramite POST (AWS Signature (versione 4))

Amazon S3 supporta le richieste HTTP POST in modo che gli utenti possano caricare contenuti direttamente su Amazon S3. L'uso di HTTP POST per caricare contenuti semplifica i caricamenti e riduce la latenza di caricamento nel momento in cui gli utenti caricano i dati da archiviare in Amazon S3. Questa sezione descrive come autenticare le richieste HTTP POST. Per ulteriori informazioni sulle richieste HTTP POST, su come creare un modulo, creare una politica POST e un esempio, vedere [Browser-Based Caricamenti tramite POST \(AWS Signature \(versione 4\)\)](#).

Per autenticare una richiesta HTTP POST, procedi come segue:

1. Il modulo deve includere i seguenti campi per fornire la firma e le informazioni pertinenti che Amazon S3 può utilizzare per ricalcolare la firma alla ricezione della richiesta:

Nome elemento	Description
policy	La politica Base64-encoded di sicurezza che descrive cosa è consentito nella richiesta. Per il calcolo della firma, questa politica è la stringa da firmare. Amazon S3 deve ricevere questa policy per poter ricalcolare la firma.
x-amz-algorithm	L'algoritmo di firma utilizzato. Per AWS Signature Version 4, il valore è <code>AWS4-HMAC-SHA256</code> .
x-amz-credential	Oltre all'ID della chiave di accesso, fornisce le informazioni sull'ambito utilizzate nel calcolo della chiave di firma per il calcolo della firma. Si tratta di una stringa del seguente formato: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Ad esempio: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</pre> Per Amazon S3, la stringa <code>aws-service</code> è <code>s3</code>. Per un elenco delle <code>aws-region</code> stringhe Amazon S3, consulta Regioni ed endpoint in. Riferimenti generali di AWS

Nome elemento	Description
x-amz-date	È il valore della data in formato ISO8601. Ad esempio, 20130728T000000Z . È la stessa data utilizzata per creare la chiave di firma. Inoltre, deve essere lo stesso valore fornito nella polizza (x-amz-date) che hai firmato.
x-amz-signature	(AWS Signature Version 4) L' HMAC-SHA256 hash della politica di sicurezza. Per ulteriori informazioni sulle opzioni per la firma, consulta Aggiungere la firma alla richiesta HTTP nel Riferimenti generali di AWS.

2. La politica POST deve includere i seguenti elementi:

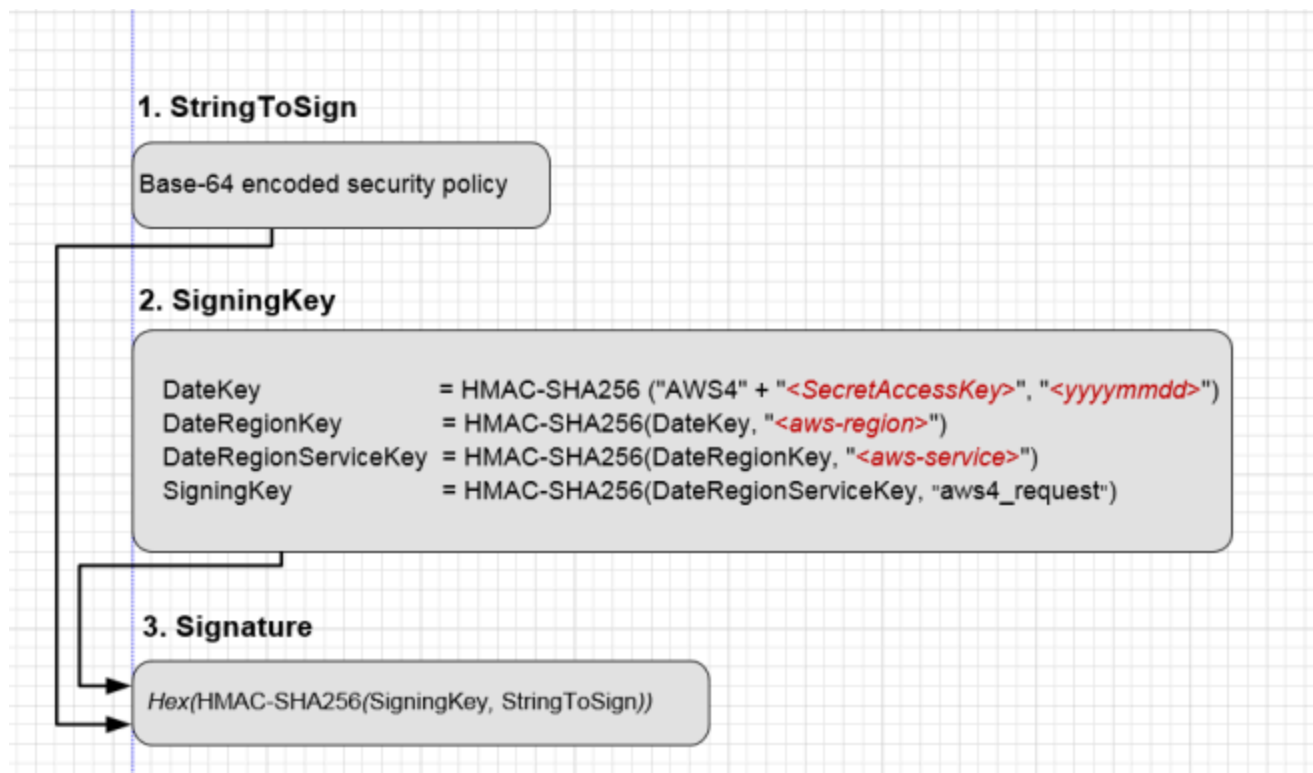
Nome elemento	Description
x-amz-algorithm	L'algoritmo di firma utilizzato per calcolare la firma. Per AWS Signature Version 4, il valore èAWS4-HMAC-SHA256.
x-amz-credential	Oltre all'ID della chiave di accesso, fornisce le informazioni sull'ambito utilizzate nel calcolo della chiave di firma per il calcolo della firma. Si tratta di una stringa del seguente formato: <i><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</i> Ad esempio, AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . .

Nome elemento	Description
x-amz-date	Il valore della data specificato nella stringa in formato ISO8601. Ad esempio, «20130728T000000Z». La data deve essere la stessa utilizzata nella creazione della chiave di firma per il calcolo della firma.

3. Per il calcolo della firma, la politica POST è la stringa da firmare.

Calcolo di una firma

Il diagramma seguente illustra il processo di calcolo della firma.



Per calcolare una firma

1. Crea una politica utilizzando la UTF-8 codifica.
2. Convertire la UTF-8-encoded policy in Base64. Il risultato è la stringa da firmare.

3. Crea la firma come HMAC-SHA256 hash della stringa da firmare. Fornirai la chiave di firma come chiave per la funzione hash.
4. Codifica la firma utilizzando la codifica esadecimale.

Per ulteriori informazioni sulla creazione di moduli HTML, politiche di sicurezza e un esempio, consultate i seguenti argomenti secondari:

- [Creazione di un modulo HTML \(utilizzando AWS Signature \(versione 4\)\)](#)
- [Politica POST](#)
- [Esempio: Browser-Based caricamento tramite HTTP POST \(utilizzando AWS Signature \(versione 4\)\)](#)

Chiavi specifiche delle policy di autenticazione di Amazon S3 Signature versione 4

La tabella seguente mostra le chiavi di policy relative all'autenticazione Amazon S3 Signature Version 4 che possono essere incluse nelle policy di Amazon S3. In una policy del bucket, puoi aggiungere queste condizioni per applicare un comportamento specifico quando le richieste vengono autenticate tramite Signature Version 4. Per esempi di policy, consulta [Esempi di policy Bucket che utilizzano le chiavi di condizione relative alla versione 4 di Signature](#).

Chiavi applicabili	Description
s3:signatureversion	Identifica la versione di AWS Signature che desideri supportare per le richieste autenticate. Per le richieste autenticate, Amazon S3 supporta sia la versione 4 di Signature che la versione 2 di Signature. Puoi aggiungere questa condizione nella tua bucket policy per richiedere una versione di firma specifica. Valori validi: "AWS" identifica la versione 2 di Signature

Chiavi applicabili	Description
	"AWS4-HMAC-SHA256" identifica la versione 4 di Signature
s3:authType	Amazon S3 supporta vari metodi di autenticazione (vedi. Autenticazione delle richieste (AWS Signature (versione 4))). Facoltativamente, puoi utilizzare questa chiave di condizione per limitare le richieste in arrivo all'uso di un metodo di autenticazione specifico. Ad esempio, puoi consentire l'utilizzo solo dell'Authorization intestazione HTTP nell'autenticazione delle richieste. Valori validi: REST-HEADER REST-QUERY-STRING POST

Chiavi applicabili	Description
s3:signatureAge	Il periodo di tempo, espresso in millisecondi, di validità di una firma in una richiesta autenticata. Questa condizione funziona per: • URL prefirmati: dove prevale la condizione e più restrittiva. Per ulteriori informazioni, consulta Lavorare con gli URL prefirmati. • POST prefirmato: carica i file direttamente su S3 utilizzando POST prefirmato. Per ulteriori informazioni, consulta la politica POST di Amazon S3. Nella versione 2 di Signature, questo valore è sempre impostato su 0. In Signature Version 4, la chiave di firma è valida per un massimo di sette giorni. Pertanto, anche le firme sono valide per un massimo di sette giorni. Puoi utilizzare questa condizione per limitare ulteriormente la durata della firma. Per ulteriori informazioni, consulta Introduzione alla firma delle richieste. Valore di esempio: 100

Chiavi applicabili	Description
s3:x-amz-content-sha256	Questa chiave di condizione può essere utilizzata per non consentire contenuti non firmati nel bucket. Quando utilizzi Signature Version 4, per le richieste che utilizzano l'intestazione <code>Authorization</code>, aggiungi l'intestazione <code>x-amz-content-sha256</code> nel calcolo della firma e quindi imposti il suo valore sul payload hash. Nota che questa chiave di condizione non supporta l'<code>x-amz-content-sha256</code> intestazione come parametro della stringa di query. Puoi utilizzare questa chiave di condizione nella tua bucket policy per negare qualsiasi caricamento in cui i payload non sono firmati. Ad esempio, puoi negare i caricamenti che utilizzano l'<code>Authorization</code> intestazione per autenticare le richieste ma non firmano il payload. Per ulteriori informazioni, consulta Calcolo delle firme per l'intestazione di autorizzazione: trasferimento del payload in un unico blocco (AWS Signature (versione 4)). Valore valido: UNSIGNED-PAYLOAD

Esempi di policy Bucket che utilizzano le chiavi di condizione relative alla versione 4 di Signature

La seguente policy sui bucket nega qualsiasi richiesta di URL prefirmato di Amazon S3 sugli oggetti `examplebucket` se la firma risale a più di dieci minuti.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Deny a presigned URL request if the signature is more than 10 min old",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::examplebucket3/*",
      "Condition": {
        "NumericGreaterThan": {
          "s3:signatureAge": 600000
        }
      }
    }
  ]
}
```

La seguente policy del bucket consente solo le richieste che utilizzano l'intestazione `Authorization` per l'autenticazione della richiesta. Qualsiasi richiesta POST o URL prefirmata verrà rifiutata.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow only requests that use Authorization header for request authentication. Deny POST or presigned URL requests.",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::examplebucket3/*",
      "Condition": {
        "StringNotEquals": {
          "s3:authType": "REST-HEADER"
        }
      }
    }
  ]
}
```

```

    }
  }
]
}

```

La seguente policy sui bucket nega le richieste che utilizzano URL prefirmati per l'autenticazione delle richieste:

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyUploadsUsingPresignedURL",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
      "Condition": {
        "StringEquals": {
          "s3:authType": "REST-query-string"
        }
      }
    }
  ]
}

```

La seguente policy sui bucket nega qualsiasi caricamento con payload non firmati:

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Deny uploads with unsigned payloads that use the  
Authorization header.",

```

```
    "Effect": "Deny",
    "Principal": "*",
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::examplebucket3/*",
    "Condition": {
      "StringEquals": {
        "s3:x-amz-content-sha256": "UNSIGNED-PAYLOAD"
      }
    }
  }
]
```

Browser-Based Caricamenti tramite POST (AWS Signature (versione 4))

Questa sezione illustra come caricare file direttamente su Amazon S3 tramite un browser utilizzando le richieste HTTP POST. Contiene inoltre informazioni su come utilizzare la JavaScript libreria AWS Amplify per il caricamento di file tramite browser su Amazon S3.

Argomenti

- [POST Object](#)
- [POST Object restore](#)
- [Browser-Based Caricamenti tramite HTTP POST](#)
- [Calcolo di una firma](#)
- [Creazione di un modulo HTML \(utilizzando AWS Signature \(versione 4\)\)](#)
- [Politica POST](#)
- [Esempio: Browser-Based caricamento tramite HTTP POST \(utilizzando AWS Signature \(versione 4\)\)](#)
- [Browser-based vengono caricati su Amazon S3 utilizzando AWS Libreria Amplify](#)

POST Object

Description

L'POSToperazione aggiunge un oggetto a un bucket specificato utilizzando moduli HTML. POSTè una forma alternativa PUT che consente i caricamenti basati su browser per inserire oggetti nei bucket. I parametri passati PUT tramite intestazioni HTTP vengono invece passati come campi del modulo POST nel corpo del messaggio con codifica -data. multipart/form Per aggiungere un oggetto a un bucket, devi avere WRITE accesso al bucket. Amazon S3 non archivia mai oggetti parziali. Se ricevi una risposta positiva, puoi essere certo che l'intero oggetto è stato archiviato.

Amazon S3 è un sistema distribuito. A meno che tu non abbia abilitato il controllo delle versioni per un bucket, se Amazon S3 riceve più richieste di scrittura per lo stesso oggetto contemporaneamente, viene archiviata solo l'ultima versione dell'oggetto scritto.

Per assicurarti che i dati non vengano danneggiati durante l'attraversamento della rete, utilizza il campo del modulo. Content-MD5 Quando utilizzi questo campo del modulo, Amazon S3 confronta l'oggetto con il valore MD5 fornito. Se non corrispondono, Amazon S3 restituisce un errore. Inoltre, puoi calcolare il valore MD5 mentre pubblichi un oggetto su Amazon S3 e confrontare il valore MD5 restituito ETag con il valore MD5 calcolato. L'ETag riflette solo le modifiche al contenuto di un oggetto, non i suoi metadati.

Note

Per configurare l'applicazione in modo che invii le intestazioni della richiesta prima di inviare il corpo della richiesta, utilizza il codice di stato HTTP 100 (Continua). Per quanto riguarda POST le operazioni, l'utilizzo di questo codice di stato consente di evitare l'invio del corpo del messaggio se il messaggio viene rifiutato in base alle intestazioni (ad esempio, a causa di un errore di autenticazione o di un reindirizzamento). Per ulteriori informazioni sul codice di stato HTTP 100 (Continua), vai alla Sezione 8.2.3 di. <http://www.ietf.org/rfc/rfc2616.txt>

Amazon S3 esegue automaticamente la crittografia di tutti i nuovi oggetti caricati in un bucket S3. L'impostazione di crittografia di un oggetto caricato dipende dalla configurazione di crittografia predefinita del bucket di destinazione. Per impostazione predefinita, tutti i bucket hanno una configurazione di crittografia predefinita che utilizza la crittografia lato server con chiavi gestite da Amazon S3 (). SSE-S3

Se il bucket di destinazione ha una configurazione di crittografia che utilizza la crittografia lato server con una AWS Key Management Service chiave (SSE-KMS), la crittografia lato server a doppio livello con una AWS KMS chiave (DSSE-KMS) o una chiave di crittografia fornita dal cliente (SSE-C), Amazon S3 utilizza la chiave KMS o la chiave fornita dal cliente corrispondente per crittografare l'oggetto caricato. AWS KMS Quando si carica un oggetto, se si desidera modificare l'impostazione di crittografia dell'oggetto caricato, è possibile specificare il tipo di crittografia lato server. È possibile configurare SSE-S3, SSE-KMS, DSSE-KMS o SSE-C Per ulteriori informazioni, consulta [Protezione dei dati con la crittografia lato server](#) nella Guida per l'utente di Amazon Simple Storage Service.

Important

Quando crei la tua richiesta, assicurati che il file campo sia l'ultimo campo del modulo.

Controllo delle versioni

Se abiliti il controllo delle versioni per un bucket, genera POST automaticamente un ID di versione univoco per l'oggetto da aggiungere. Amazon S3 restituisce questo ID nella risposta utilizzando l'intestazione della `x-amz-version-id` risposta.

Se sospendi il controllo delle versioni per un bucket, Amazon S3 utilizza sempre `null` come ID di versione dell'oggetto archiviato in un bucket.

Per ulteriori informazioni sulla restituzione dello stato di controllo delle versioni di un bucket, consulta [GetBucketVersioning](#)

Amazon S3 è un sistema distribuito. Se abiliti il controllo delle versioni per un bucket e Amazon S3 riceve più richieste di scrittura per lo stesso oggetto contemporaneamente, tutte le versioni dell'oggetto vengono archiviate.

Per visualizzare esempi di richieste che utilizzano il controllo delle versioni, consulta Sample Request. [???](#)

Richieste

Sintassi

```
POST / HTTP/1.1
Host: destinationBucket.s3.amazonaws.com
User-Agent: browser_data
```

```
Accept: file_types
Accept-Language: Regions
Accept-Encoding: encoding
Accept-Charset: character_set
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=9431149156168
Content-Length: length

--9431149156168
Content-Disposition: form-data; name="key"

acl
--9431149156168
Content-Disposition: form-data; name="tagging"

<Tagging><TagSet><Tag><Key>Tag Name</Key><Value>Tag Value</Value></Tag></TagSet></
Tagging>
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"

success_redirect
--9431149156168
Content-Disposition: form-data; name="Content-Type"

content_type
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-uuid"

uuid
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-tag"

metadata
--9431149156168
Content-Disposition: form-data; name="AWSAccessKeyId"

access-key-id
--9431149156168
Content-Disposition: form-data; name="Policy"

encoded_policy
--9431149156168
Content-Disposition: form-data; name="Signature"
```

```
signature=
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
Content-Type: image/jpeg

file_content
--9431149156168
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--9431149156168--
```

Parametri della richiesta

Questa implementazione dell'operazione non utilizza parametri di richiesta.

Campi del modulo

Questa operazione può utilizzare i seguenti campi del modulo.

Nome	Description	Richiesto
AWSAccessKeyId	L'ID della chiave di AWS accesso del proprietario del bucket che concede a un utente anonimo l'accesso per una richiesta che soddisfa il set di vincoli della policy. Tipo: stringa Impostazione predefinita: nessuna Vincoli: obbligatorio se nella richiesta è incluso un documento di policy.	Condizionale
acl	L'elenco di controllo degli accessi (ACL) di Amazon S3 specificato. Se l'ACL specificato non è valido, viene generato un errore. Per ulteriori informazioni sugli ACL, consulta la panoramica degli elenchi di controllo degli accessi (ACL) nella Amazon Simple Storage Service User Guide.	No

Nome	Description	Richiesto
	Tipo: stringa Default: private Valori validi: private public-read public-read-write aws-exec-read authenticated-read bucket-owner-read bucket-owner-full-control	
Cache-Control , Content-Type , Content-Disposition , Content-Encoding , Expires	Le REST-specific intestazioni. Per ulteriori informazioni, consulta PutObject. Tipo: stringa Impostazione predefinita: nessuna	No
file	Il contenuto del file o del testo. Il contenuto del file o del testo deve essere l'ultimo campo del modulo. Non è possibile caricare più di un file alla volta. Tipo: contenuto del file o del testo Impostazione predefinita: nessuna	Sì

Nome	Description	Richiesto
key	Il nome della chiave caricata. Per utilizzare il nome di file fornito dall'utente, utilizzare la <code>\${filename}</code> variabile. Ad esempio, se un utente di nome Mary carica il file <code>example.jpg</code> e tu lo specifichi <code>/user/mary/\${filename}</code>, il nome della chiave è <code>/user/mary/example.jpg</code>. Per ulteriori informazioni, consulta la chiave dell'oggetto e i metadati nella <i>Amazon Simple Storage Service User Guide</i>. Tipo: stringa Impostazione predefinita: nessuna	Sì
policy	La politica di sicurezza che descrive cosa è consentito nella richiesta. Le richieste senza una politica di sicurezza sono considerate anonime e funzionano solo su bucket scrivibili pubblicamente. Per ulteriori informazioni, consulta i moduli HTML e gli esempi di caricamento nella <i>Amazon Simple Storage Service User Guide</i>. Tipo: stringa Impostazione predefinita: nessuna Vincoli: è necessaria una policy di sicurezza se il bucket non è scrivibile pubblicamente.	Condizionale

Nome	Description	Richiesto
<code>success_action_redirect</code> , <code>redirect</code>	L'URL a cui il client viene reindirizzato dopo un caricamento riuscito. Se <code>success_action_redirect</code> non è specificato, Amazon S3 restituisce il tipo di documento vuoto specificato nel campo <code>success_action_status</code>. Se Amazon S3 non è in grado di interpretare l'URL, si comporta come se il campo non fosse presente. Se il caricamento non riesce, Amazon S3 visualizza un errore e non reindirizza l'utente a un URL. Tipo: stringa Impostazione predefinita: nessuna  Note Il nome del <code>redirect</code> campo è obsoleto e il supporto per il nome del <code>redirect</code> campo verrà rimosso in futuro.	No

Nome	Description	Richiesto
success_action_status	Se non lo specifichi <code>success_action_redirect</code>, il codice di stato viene restituito al client quando il caricamento ha esito positivo. Questo campo accetta i valori <code>200</code>, <code>201</code>, o <code>204</code> (impostazione predefinita). Se il valore è impostato su <code>200</code> o <code>204</code>, Amazon S3 restituisce un documento vuoto con un codice di stato <code>200</code> o <code>204</code>. Se il valore è impostato su <code>201</code>, Amazon S3 restituisce un documento XML con un codice di stato <code>201</code>. Se il valore non è impostato o se è impostato su un valore non valido, Amazon S3 restituisce un documento vuoto con un codice di stato <code>204</code>. Tipo: stringa Impostazione predefinita: nessuna	No

Nome	Description	Richiesto
tagging	Il set di tag specificato da aggiungere all'oggetto. Per aggiungere tag, utilizzate il seguente schema di codifica. <pre><Tagging> <TagSet> <Tag> <Key>TagName</Key> <Value>TagValue</Value> </Tag> ... </TagSet> </Tagging></pre> Per ulteriori informazioni, consulta l'etichettatura degli oggetti nella Amazon Simple Storage Service User Guide. Tipo: stringa Impostazione predefinita: nessuna	No

Nome	Description	Richiesto
<code>x-amz-storage-class</code>	La classe di storage da utilizzare per archiviare l'oggetto. Se non si specifica una classe, Amazon S3 utilizza la classe di storage predefinita, STANDARD. Amazon S3 supporta altre classi di storage. Per ulteriori informazioni, consulta la sezione Storage classes nella Amazon Simple Storage Service User Guide. Tipo: stringa Impostazione predefinita: STANDARD Valori validi: REDUCED_REDUNDANCY EXPRESS_ONEZONE DEEP_ARCHIVE GLACIER GLACIER_IR INTELLIGENT_TIERING ONEZONE_IA STANDARD STANDARD_IA	No
<code>x-amz-meta-*</code>	Le intestazioni che iniziano con questo prefisso sono metadati definiti dall'utente. Ognuna viene archiviata e restituita come set di coppie chiave-valore. Amazon S3 non convalida né interpreta i metadati definiti dall'utente. Per ulteriori informazioni, consulta PutObject. Tipo: stringa Impostazione predefinita: nessuna	No

Nome	Description	Richiesto
x-amz-security-token	Il token di sicurezza Amazon. DevPay Ogni richiesta che utilizza Amazon DevPay richiede due campi del x-amz-security-token modulo: uno per il token del prodotto e uno per il token utente. Tipo: stringa Impostazione predefinita: nessuna	No
x-amz-signature	(AWS Signature Version 4) L' HMAC-SHA256 hash della politica di sicurezza. Tipo: stringa Impostazione predefinita: nessuna	Condizionale

Nome	Description	Richiesto
x-amz-website-redirect-location	Se il bucket è configurato come sito Web, questo campo reindirizza le richieste per questo oggetto a un altro oggetto nello stesso bucket o a un URL esterno. Amazon S3 memorizza il valore di questa intestazione nei metadati dell'oggetto. Per informazioni sui metadati degli oggetti, consulta la chiave e i metadati dell'oggetto nella Amazon Simple Storage Service User Guide. Nell'esempio seguente, l'intestazione della richiesta imposta il reindirizzamento a un object (<code>anotherPage.html</code>) nello stesso bucket: <pre>x-amz-website-redirect-location: /anotherPage.html</pre> Nell'esempio seguente, l'intestazione della richiesta imposta il reindirizzamento dell'oggetto a un altro sito Web: <pre>x-amz-website-redirect-location: http://www.example.com/</pre> Per ulteriori informazioni sull'hosting di siti Web in Amazon S3, consulta Hosting websites on Amazon S3 e Come configurare i reindirizzamenti delle pagine dei siti Web nella Amazon Simple Storage Service User Guide. Tipo: stringa Impostazione predefinita: nessuna Vincoli: il valore deve essere preceduto da,, o. / http:// https:// La lunghezza del valore è limitata a 2 KB.	No

Campi aggiuntivi del modulo di richiesta di checksum

Quando carichi un oggetto, puoi specificare vari checksum da utilizzare per verificare l'integrità dei dati. Puoi specificare un algoritmo di checksum aggiuntivo da utilizzare per Amazon S3. Per ulteriori informazioni sui valori di checksum aggiuntivi, consulta [Verifica dell'integrità degli oggetti](#) nella Amazon Simple Storage Service User Guide.

Nome	Description	Richiesto
x-amz-checksum-algorithm	Indica l'algoritmo utilizzato per creare il checksum per l'oggetto. Se viene specificato un valore, è necessario includere l'intestazione di checksum corrispondente. In caso contrario, la richiesta genererà un errore 400. I valori possibili includono CRC32, CRC32C, SHA1 e SHA256.	No
x-amz-checksum-crc32	Specifica il checksum CRC32 a 32 bit codificato in base64 dell'oggetto. Questo parametro è obbligatorio se il valore di è. x-amz-checksum-algorithm CRC32	Condizionale
x-amz-checksum-crc32c	Specifica il checksum CRC32C a 32 bit con codifica base64 dell'oggetto. Questo parametro è obbligatorio se il valore di è. x-amz-checksum-algorithm CRC32C	Condizionale
x-amz-checksum-sha1	Specifica il digest a 160 SHA-1 bit con codifica base64 dell'oggetto. Questo parametro è obbligatorio se il valore di è. x-amz-checksum-algorithm SHA1	Condizionale
x-amz-checksum-sha256	Specifica il digest a 256 SHA-256 bit con codifica base64 dell'oggetto. Questo parametro è obbligatorio se il valore di è. x-amz-checksum-algorithm SHA256	Condizionale

Server-Side Campi specifici del modulo di richiesta di crittografia

Server-side la crittografia è la crittografia dei dati inattivi. Amazon S3 crittografa i tuoi dati durante la scrittura su dischi nei data AWS center e li decrittografa quando accedi ad essi. Quando carichi un oggetto, puoi specificare il tipo di crittografia lato server che desideri che Amazon S3 utilizzi per crittografare l'oggetto.

Esistono quattro tipi di crittografia lato server:

- Server-side crittografia con chiavi gestite Amazon S3 (SSE-S3): a partire da maggio 2022, tutti i bucket Amazon S3 hanno la crittografia configurata per impostazione predefinita. L'opzione predefinita per la crittografia lato server è with. SSE-S3 Ogni oggetto è crittografato con una chiave univoca. Come protezione aggiuntiva, SSE-S3 crittografa la chiave stessa con una chiave radice che ruota regolarmente. SSE-S3 utilizza uno dei più potenti cifrari a blocchi disponibili, Advanced Encryption Standard () a 256 bit, per crittografare i dati. AES-256
- Server-side crittografia con AWS KMS chiavi (SSE-KMS) — SSE-KMS viene fornita tramite un'integrazione del AWS KMS servizio con Amazon S3. Con AWS KMS, hai un maggiore controllo sulle tue chiavi. Ad esempio, puoi visualizzare le chiavi separate, modificare le policy di controllo e seguire le chiavi in AWS CloudTrail. Inoltre, puoi creare e gestire chiavi gestite dal cliente oppure utilizzare chiavi Chiavi gestite da AWS create appositamente per te, il tuo servizio e la tua regione.
- Dual-layer crittografia lato server con AWS KMS chiavi (DSSE-KMS): la Dual-layer crittografia lato server con AWS KMS keys (DSSE-KMS) è simile a SSE-KMS, ma applica due singoli livelli di crittografia a livello di oggetto anziché un livello.
- Server-side crittografia con chiavi fornite dal cliente (SSE-C): con SSE-C, gestisci le chiavi di crittografia e Amazon S3 gestisce la crittografia durante la scrittura sui dischi e la decrittografia quando accedi ai tuoi oggetti.

Note

Se la crittografia lato server con le chiavi fornite dal cliente (SSE-C) è bloccata per il bucket generico, riceverai un errore HTTP 403 Access Denied quando specifichi le intestazioni della richiesta durante la scrittura di nuovi dati nel bucket. SSE-C Per ulteriori informazioni, consulta [Bloccare o sbloccare un bucket per scopi generici. SSE-C](#)

Per ulteriori informazioni, consulta [Protecting data using server-side encryption](#) nella Amazon Simple Storage Service User Guide.

A seconda del tipo di crittografia lato server che desideri utilizzare, specifica i seguenti campi del modulo.

- Usa SSE-S3 SSE-KMS, o DSSE-KMS: se desideri utilizzare questi tipi di crittografia lato server, specifica i seguenti campi del modulo nella richiesta.

Nome	Description	Richiesto
<code>x-amz-server-side-encryption</code>	Specifica l'algoritmo di crittografia lato server da utilizzare e quando Amazon S3 crea un oggetto. Per utilizzarlo, specifica. SSE-S3 AES256 Per usare SSE-KMS, specifica <code>aws:kms</code>. Per usare DSSE-KMS, specifica <code>aws:kms:dsse</code>. Tipo: stringa Valore valido: <code>aws:kms</code>, <code>AES256</code>, <code>aws:kms:dsse</code>	Sì
<code>x-amz-server-side-encryption-aws-kms-key-id</code>	Se l'<code>x-amz-server-side-encryption</code> intestazione ha un valore valido pari a <code>aws:kms</code> o <code>aws:kms:dsse</code>, questa intestazione specifica l'ID utilizzato per crittografare l'AWS KMS key oggetto. Tipo: stringa	Sì, se il valore di <code>x-amz-server-side-encryption</code> è <code>aws:kms</code> o <code>aws:kms:dsse</code>
<code>x-amz-server-side-encryption-context</code>	Se <code>x-amz-server-side-encryption</code> il valore valido è <code>aws:kms</code> o <code>aws:kms:dsse</code>, questa intestazione specifica il contesto di crittografia per l'oggetto. Il valore di questa intestazione è una UTF-8 stringa con codifica Base64 che contiene JSON-formatted coppie chiave-valore per il contesto di crittografia. Tipo: stringa	No

Nome	Description	Richiesto
x-amz-server-side-encryption-bucket-key-enabled	Se il valore valido x-amz-server-side-encryption è aws:kms o aws:kms:dsse, questa intestazione specifica se Amazon S3 deve utilizzare una chiave S3 Bucket con o. SSE-KMS DSSE-KMS Se si imposta questa intestazione su, Amazon S3 utilizza una chiave bucket S3 per la crittografia degli oggetti con o. true SSE-KMS DSSE-KMS Tipo: Booleano	No

 Note

Se specifichi x-amz-server-side-encryption:aws:kms o x-amz-server-side-encryption:aws:kms:dsse, ma non lo fornisci x-amz-server-side-encryption-aws-kms-key-id, Amazon S3 utilizza Chiave gestita da AWS (aws/S3) per proteggere i dati.

- Uso SSE-C: se desideri gestire le tue chiavi di crittografia, devi fornire tutti i seguenti campi del modulo nella richiesta.

 Note

Se lo usi SSE-C, il ETag valore che Amazon S3 restituisce nella risposta non è l'MD5 dell'oggetto.

Nome	Description	Richiesto
x-amz-server-side-encryption-customer-algorithm	Specifica l'algoritmo da utilizzare per crittografare l'oggetto. Tipo: stringa Impostazione predefinita: nessuna Valore valido: AES256	Sì

Nome	Description	Richiesto
	Vincoli: deve essere accompagnato da <code>x-amz-server-side-encryption-customer-key-MD5</code> e campi <code>x-amz-server-side-encryption-customer-key</code> e validi.	
<code>x-amz-server-side-encryption-customer-key</code>	Specifica la chiave di crittografia con codifica Base64 fornita dal cliente per Amazon S3 da utilizzare nella crittografia dei dati. Questo valore viene utilizzato per memorizzare l'oggetto e quindi viene eliminato. Amazon non memorizza la chiave di crittografia. La chiave deve essere appropriata all'uso con l'algoritmo specificato nell'<code>x-amz-server-side-encryption-customer-algorithm</code> intestazione. Tipo: stringa Impostazione predefinita: nessuna Vincoli: deve essere accompagnato da campi <code>x-amz-server-side-encryption-customer-algorithm</code> e <code>x-amz-server-side-encryption-customer-key-MD5</code> validi.	Sì

Nome	Description	Richiesto
x-amz-server-side-encryption-customer-key-MD5	Specifica il digest MD5 a 128 bit con codifica base64 della chiave di crittografia secondo RFC 1321. http://tools.ietf.org/html/rfc1321 Amazon S3 utilizza questa intestazione per un controllo dell'integrità dei messaggi per garantire che la chiave di crittografia sia stata trasmessa senza errori. Tipo: stringa Impostazione predefinita: nessuna Vincoli: deve essere accompagnato da campi e validi. x-amz-server-side-encryption-customer-algorithm x-amz-server-side-encryption-customer-key	Sì

Risposte

Intestazioni di risposta

Questa implementazione dell'operazione può includere le seguenti intestazioni di risposta oltre alle intestazioni di risposta comuni a tutte le risposte. Per ulteriori informazioni, vedere [Common Response Headers](#).

Nome	Description
x-amz-checksum-crc32	Il checksum CRC32 a 32 bit con codifica base64 dell'oggetto. Tipo: stringa
x-amz-checksum-crc32c	Il checksum CRC32C a 32 bit con codifica base64 dell'oggetto. Tipo: stringa
x-amz-checksum-sha1	Il digest a 160 bit dell'oggetto con codifica in base64. SHA-1

Nome	Description
	Tipo: stringa
x-amz-checksum-sha256	Digest a 256 bit dell'oggetto con codifica in base64. SHA-256 Tipo: stringa
x-amz-expiration	Se un'Expiration azione è configurata per l'oggetto come parte della configurazione del ciclo di vita del bucket, Amazon S3 restituisce questa intestazione. Il valore dell'intestazione include un componente e un componente. expiry-date URL-encoded rule-id Per i bucket abilitati alla versione, questa intestazione si applica solo alle versioni correnti. Amazon S3 non fornisce un'intestazione per indicare quando una versione non corrente è idonea per l'eliminazione permanente. Per ulteriori informazioni, consulta PutBucketLifecycleConfiguration . Tipo: stringa
success_action_redirect, redirect	L'URL a cui il client viene reindirizzato in caso di caricamento riuscito. Tipo: stringa Antenato: PostResponse
x-amz-server-side-encryption	L'algoritmo di crittografia lato server utilizzato per archiviare questo oggetto in Amazon S3 (ad esempio,, AES256). aws:kms aws:kms:dsse Tipo: stringa

Nome	Description
<code>x-amz-server-side-encryption-aws-kms-key-id</code>	Se l'<code>x-amz-server-side-encryption</code> intestazione ha un valore valido di <code>aws:kms</code>, questa intestazione specifica l'ID della chiave KMS utilizzata per crittografare l'oggetto. Tipo: stringa
<code>x-amz-server-side-encryption-bucket-key-enabled</code>	Se <code>x-amz-server-side-encryption</code> ha un valore valido di <code>aws:kms</code>, questa intestazione indica se l'oggetto è crittografato SSE-KMS utilizzando una chiave S3 Bucket. Se questa intestazione è impostata su <code>true</code>, l'oggetto utilizza una chiave S3 Bucket con SSE-KMS. Tipo: Booleano
<code>x-amz-server-side-encryption-customer-algorithm</code>	Se SSE-C richiesto, la risposta include questa intestazione, che conferma l'algoritmo di crittografia utilizzato. Tipo: stringa Valori validi: AES256
<code>x-amz-server-side-encryption-customer-key-MD5</code>	Se SSE-C richiesto, la risposta include questa intestazione per verificare l'integrità del messaggio di andata e ritorno della chiave di crittografia fornita dal cliente. Tipo: stringa
<code>x-amz-version-id</code>	Versione dell'oggetto. Tipo: stringa

Elementi di risposta

Nome	Description
Bucket	Il nome del bucket in cui è stato archiviato l'oggetto. Tipo: stringa Antenato: PostResponse
ETag	L'entity tag (ETag) è un hash MD5 dell'oggetto che è possibile utilizzare per eseguire GET operazioni condizionali utilizzando il tag di richiesta con l'operazione di If-Modified richiesta . GET ETagriflette le modifiche solo al contenuto di un oggetto, non ai suoi metadati. Tipo: stringa Antenato: PostResponse
Key	Il nome della chiave dell'oggetto. Tipo: stringa Antenato: PostResponse
Location	L'URI dell'oggetto. Tipo: stringa Antenato: PostResponse

Errori speciali

Questa implementazione dell'operazione non restituisce errori speciali. Per informazioni generali sugli errori di Amazon S3 e un elenco di codici di errore, consulta [Error Responses](#).

Esempi

Richiesta di esempio

```
POST /Neo HTTP/1.1
Content-Length: 4
Host: quotes.s3.amazonaws.com
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: authorization string
Content-Type: text/plain
Expect: the 100-continue HTTP status code
```

ObjectContent

Esempio di risposta con controllo delle versioni sospeso

Di seguito è riportato un esempio di risposta quando il controllo delle versioni del bucket è sospeso:

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
x-amz-id-2: LriYPLdm0dAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiI0Ix1+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: default
Date: Wed, 12 Oct 2009 17:50:00 GMT
ETag: "1b2cf535f27731c974343645a3985328"
Content-Length: 0
Connection: close
Server: AmazonS3
```

In questa risposta, l'ID della versione è. null

Esempio di risposta con controllo delle versioni abilitato

Di seguito è riportato un esempio di risposta quando il controllo delle versioni del bucket è abilitato.

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
x-amz-id-2: LriYPLdm0dAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiI0Ix1+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: 43jfkodU8493jnFJD9fjj3HHNVfdsQUIFDNsidf038jfdsjGFDSIRp
Date: Wed, 01 Mar 2006 12:00:00 GMT
```

```
ETag: "828ef3fdfa96f00ad9f27c383fc9ac7f"  
Content-Length: 0  
Connection: close  
Server: AmazonS3
```

Risorse correlate

- [CopyObject](#)
- [POST Object](#)
- [GetObject](#)

POST Object restore

Description

Questa operazione esegue i seguenti tipi di richieste:

- `select`— Esegue un'interrogazione di selezione su un oggetto archiviato
- `restore an archive`— Ripristinare un oggetto archiviato

Per utilizzare questa operazione, è necessario disporre delle autorizzazioni per eseguire le azioni `s3:RestoreObject` e `s3:GetObject`. Il proprietario del bucket dispone di questa autorizzazione per impostazione predefinita e può concedere questa autorizzazione ad altri. Per ulteriori informazioni sulle autorizzazioni, consulta [Autorizzazioni relative alle operazioni sulle subrisorse del bucket](#) [e Gestione delle autorizzazioni di accesso alle risorse Amazon S3](#) nella Guida per l'utente di Amazon Simple Storage Service.

Interrogazione degli archivi con Select Requests

Si utilizza un tipo di richiesta selezionato per eseguire interrogazioni SQL sugli oggetti archiviati. Gli oggetti archiviati interrogati dalla richiesta di selezione devono essere formattati come file CSV (valori separati da virgole) non compressi. Puoi eseguire query e analisi personalizzate sui dati archiviati senza doverli ripristinare su un livello Amazon S3 più caldo. Per una panoramica sulle richieste selezionate, consulta [Querying Archived Objects](#) nella Amazon Simple Storage Service User Guide.

Quando effettui una richiesta di selezione, procedi come segue:

- Definisci una posizione di output per l'output della query di selezione. Deve essere un bucket Amazon S3 nello stesso Regione AWS bucket che contiene l'oggetto di archivio su cui viene interrogato. Il programma Account AWS che avvia il processo deve disporre delle autorizzazioni per scrivere nel bucket S3. È possibile specificare la classe di archiviazione e la crittografia per gli oggetti di output archiviati nel bucket. Per ulteriori informazioni sull'output, consulta [Querying Archived Objects](#) nella Amazon Simple Storage Service User Guide.

Per ulteriori informazioni sulla S3 struttura nel corpo della richiesta, consulta quanto segue:

- [PutObject](#)
- [Gestire l'accesso con gli ACL](#) nella Amazon Simple Storage Service User Guide

- [Protezione dei dati mediante Server-Side crittografia](#) nella Guida per l'utente di Amazon Simple Storage Service
- Definisci l'espressione SQL per il SELECT tipo di ripristino della tua query nella `SelectParameters` struttura del corpo della richiesta. È possibile utilizzare espressioni come negli esempi seguenti.
- L'espressione seguente restituisce tutti i record dell'oggetto specificato.

```
SELECT * FROM Object
```

- Supponendo che non si stia utilizzando alcuna intestazione per i dati archiviati nell'oggetto, è possibile specificare colonne con intestazioni posizionali.

```
SELECT s._1, s._2 FROM Object s WHERE s._3 > 100
```

- Se si dispone di intestazioni e le si imposta nella CSV struttura `fileHeaderInfo` nel corpo della richiesta su `USE`, è possibile specificare le intestazioni nella query. (se imposti il campo `fileHeaderInfo` su `IGNORE`, la prima riga viene ignorata per la query). Non puoi combinare posizioni ordinali con i nomi di colonna delle intestazioni.

```
SELECT s.Id, s.FirstName, s.SSN FROM S3Object s
```

Per ulteriori informazioni sull'utilizzo di SQL con Amazon Glacier Select restore, consulta [SQL Reference for Amazon S3 Select e Amazon Glacier Select](#) nella Amazon Simple Storage Service User Guide.

Quando effettui una richiesta di selezione, puoi anche fare quanto segue:

- Per velocizzare le tue domande, specifica il livello. `Expedited` Per ulteriori informazioni sui livelli, vedere «Ripristino degli archivi» più avanti in questo argomento.
- Specificate i dettagli sul formato di serializzazione dei dati sia dell'oggetto di input che viene interrogato sia sulla serializzazione dei risultati della query. `CSV-encoded`

Di seguito sono riportati altri dati importanti sulla funzionalità di selezione:

- I risultati in uscita sono nuovi oggetti Amazon S3. A differenza dei recuperi dagli archivi, vengono archiviati fino a quando non vengono eliminati esplicitamente, manualmente o tramite una politica del ciclo di vita.

- Puoi inviare più di una richiesta di selezione sullo stesso oggetto Amazon S3. Amazon S3 non deduplica le richieste, quindi evita di emettere richieste duplicate.
- Amazon S3 accetta una richiesta di selezione anche se l'oggetto è già stato ripristinato. Una richiesta di selezione non restituisce una risposta 409 di errore.

Ripristino degli archivi

Gli oggetti nelle classi di archiviazione GLACIER e DEEP_ARCHIVE vengono archiviati. Per accedere a un oggetto archiviato, è necessario innanzitutto avviare una richiesta di ripristino. In questo modo viene ripristinata una copia temporanea dell'oggetto archiviato. In una richiesta di ripristino, si specifica il numero di giorni in cui si desidera che la copia ripristinata esista. Dopo il periodo specificato, Amazon S3 elimina la copia temporanea ma l'oggetto rimane archiviato nella classe di storage GLACIER o DEEP_ARCHIVE da cui è stato ripristinato l'oggetto.

Per ripristinare una versione specifica dell'oggetto, puoi fornire un ID di versione. Se non fornisci un ID di versione, Amazon S3 ripristina la versione corrente.

Il tempo necessario per completare i processi di ripristino dipende dalla classe di storage da cui viene ripristinato l'oggetto e dal livello di accesso ai dati specificato.

Quando si ripristina un oggetto archiviato (o si utilizza una richiesta di selezione), è possibile specificare una delle seguenti opzioni di livello di accesso ai dati nell'elemento del corpo della richiesta:

- **Expedited**- I recuperi rapidi consentono di accedere rapidamente ai dati archiviati nella classe di archiviazione GLACIER quando sono richieste urgenti occasionali per un sottoinsieme di archivi. Per tutti gli oggetti archiviati tranne quelli più grandi (oltre 250 MB), i dati a cui si accede tramite recuperi rapidi sono in genere resi disponibili entro 1-5 minuti. La capacità assegnata assicura che la capacità di recupero per ripristini di tipo Expedited sia disponibile quando richiesta. I recuperi rapidi e la capacità assegnata non sono disponibili per la classe di storage DEEP_ARCHIVE.
- **Standard** - I recuperi Standard consentono di accedere a qualsiasi oggetto archiviato entro poche ore. Questa è l'opzione predefinita per le richieste di recupero GLACIER e DEEP_ARCHIVE che non specificano l'opzione di recupero. I recuperi standard vengono in genere completati entro 3-5 ore dalla classe di archiviazione GLACIER e in genere entro 12 ore dalla classe di archiviazione DEEP_ARCHIVE.
- **Bulk**- I recuperi in blocco sono l'opzione di recupero più economica di Amazon Glacier e consentono di recuperare grandi quantità, anche petabyte, di dati in modo economico in un giorno.

I recuperi in blocco vengono generalmente completati entro 5-12 ore dalla classe di storage GLACIER e in genere entro 48 ore dalla classe di storage DEEP_ARCHIVE.

Per ulteriori informazioni sulle opzioni di recupero degli archivi e sulla capacità assegnata per l'accesso ai Expedited dati, consulta [Restoring Archived Objects nella Amazon Simple Storage Service User Guide](#).

Puoi utilizzare Amazon S3 Restore Speed Upgrade per modificare la velocità di ripristino a una velocità maggiore mentre è in corso. Puoi aumentare la velocità di un ripristino in corso inviando un'altra richiesta di ripristino sullo stesso oggetto, impostando un nuovo Tier elemento di richiesta. Quando si invia una richiesta di aggiornamento del livello di ripristino, è necessario scegliere un livello con velocità maggiore rispetto a quella del livello utilizzato dal ripristino in corso. Non devi cambiare gli altri parametri, ad esempio l'elemento della richiesta Days. Per ulteriori informazioni, consulta [Upgrade the Speed of an In-Progress Restore](#) nella Amazon Simple Storage Service User Guide.

Per conoscere lo stato del ripristino degli oggetti, puoi inviare una HEAD richiesta. Le operazioni restituiscono l'`x-amz-restore` intestazione, che fornisce informazioni sullo stato del ripristino, nella risposta. Puoi utilizzare le notifiche di eventi di Amazon S3 per notificarti quando un ripristino viene avviato o completato. Per ulteriori informazioni, consulta [Configurazione delle notifiche di eventi di Amazon S3](#) nella Guida per l'utente di Amazon Simple Storage Service.

Dopo aver ripristinato un oggetto archiviato, puoi aggiornare il periodo di ripristino rimettendo la richiesta con un nuovo periodo. Amazon S3 aggiorna il periodo di ripristino rispetto all'ora corrente e addebita solo la richiesta, senza costi di trasferimento dei dati. Non è possibile aggiornare il periodo di ripristino quando Amazon S3 sta elaborando attivamente la richiesta di ripristino corrente per l'oggetto.

Se il bucket ha una configurazione del ciclo di vita con una regola che include un'azione di scadenza, la scadenza dell'oggetto ha la precedenza sulla durata specificata in una richiesta di ripristino. Ad esempio, se ripristini una copia dell'oggetto per 10 giorni, ma la scadenza dell'oggetto è prevista tra 3 giorni, Amazon S3 elimina l'oggetto entro 3 giorni. Per ulteriori informazioni sulla configurazione del ciclo di vita, consulta la sezione [Object Lifecycle Management](#) nella [PutBucketLifecycleConfiguration](#) Amazon Simple Storage Service User Guide.

Richieste

Sintassi

```
POST /ObjectName?restore&versionId=VersionID HTTP/1.1
```

```
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Autenticazione delle richieste \(AWS Signature \(versione 4\)\))
Content-MD5: MD5
```

request body

Note

La sintassi mostra alcune intestazioni delle richieste. Per un elenco completo, consulta «Intestazioni delle richieste» più avanti in questo argomento.

Parametri della richiesta

Questa implementazione dell'operazione non utilizza parametri di richiesta.

Intestazioni di richiesta

Nome	Description	Richiesto
Content-MD5	Il digest MD5 a 128 bit codificato in base64 dei dati. È necessario o utilizzare questa intestazione come controllo dell'integrità del messaggio per verificare che il corpo della richiesta non sia stato danneggiato durante il transito. Per ulteriori informazioni, vedere RFC 1864. Tipo: stringa Impostazione predefinita: nessuna	Sì

Elementi della richiesta

Di seguito è riportato un esempio XML del corpo di una richiesta per il ripristino di un archivio.

```
<RestoreRequest>
  <Days>2</Days>
```



```
<GlacierJobParameters>
  <Tier>Bulk</Tier>
</GlacierJobParameters>
</RestoreRequest>
```

La tabella seguente illustra il codice XML per il ripristino dell'archivio nel corpo della richiesta.

Nome	Description	Richiesto
RestoreRequest	Contenitore per il ripristino delle informazioni. Tipo: Container	Sì
Days	Durata della copia ripristinata (attiva). Il numero minimo di giorni per ripristinare un oggetto da Amazon Glacier è 1. Dopo che la copia dell'oggetto ha raggiunto la durata specificata, Amazon S3 la rimuove dal bucket. Se stai ripristinando un archivio, questo elemento è obbligatorio. Non utilizzare questo elemento con un SELECT tipo di richiesta. Tipo: numero intero positivo Antenati: RestoreRequest	Sì, se si sta ripristinando un archivio
GlacierJobParameters	Contenitore per i parametri di lavoro della classe di archiviazione S3 Glacier. Non utilizzare questo elemento con un SELECT tipo di richiesta. Tipo: Container Antenati: RestoreRequest	No
Tier	Il livello di accesso ai dati da utilizzare per il ripristino dell'archivio. Standard è il valore predefinito.	No

Nome	Description	Richiesto
	Tipo: Enum	
	Valori validi: Expedited Standard Bulk	
	Antenati: GlacierJobParameters	

Il seguente codice XML è il corpo della richiesta per un'interrogazione di selezione su un oggetto archiviato:

```
<RestoreRequest>
  <Type>SELECT</Type>
  <Tier>Expedited</Tier>
  <Description>Job description</Description>
  <SelectParameters>
    <Expression>Select * from Object</Expression>
    <ExpressionType>SQL</ExpressionType>
    <InputSerialization>
      <CSV>
        <FileHeaderInfo>IGNORE</FileHeaderInfo>
        <RecordDelimiter>\n</RecordDelimiter>
        <FieldDelimiter>,</FieldDelimiter>
        <QuoteCharacter>"</QuoteCharacter>
        <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
        <Comments>#</Comments>
      </CSV>
    </InputSerialization>
    <OutputSerialization>
      <CSV>
        <QuoteFields>ASNEEDED</QuoteFields>
        <RecordDelimiter>\n</RecordDelimiter>
        <FieldDelimiter>,</FieldDelimiter>
        <QuoteCharacter>"</QuoteCharacter>
        <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
      </CSV>
    </OutputSerialization>
  </SelectParameters>
  <OutputLocation>
    <S3>
      <BucketName>Name of bucket</BucketName>
      <Prefix>Key prefix</Prefix>
```

```

    <CannedACL>Canned ACL string</CannedACL>
    <AccessControlList>
      <Grantee>
        <Type>Grantee Type</Type>
      <ID>Grantee identifier</ID>
      <URI>Grantee URI</URI>
      <Permission>Granted permission</Permission>
      <DisplayName>Display Name</DisplayName>
      <EmailAddress>email</EmailAddress>
    </Grantee>
  </AccessControlList>
  <Encryption>
    <EncryptionType>Encryption type</EncryptionType>
    <KMSKeyId>KMS Key ID</KMSKeyId>
    <KMSContext>Base64-encoded JSON<KMSContext>
      </Encryption>
  <UserMetadata>
    <MetadataEntry>
      <Name>Key</Name>
      <Value>Value</Value>
    </MetadataEntry>
  </UserMetadata>
  <Tagging>
    <TagSet>
      <Tag>
        <Key>Tag name</Key>
        <Value>Tag value</Value>
      </Tag>
    </TagSet>
  </Tagging>
  <StorageClass>Storage class</StorageClass>
</S3>
</OutputLocation>
</RestoreRequest>

```

Le tabelle seguenti spiegano il codice XML per un SELECT tipo di ripristino nel corpo della richiesta.

Nome	Description	Richiesto
RestoreRequest	Contenitore per il ripristino delle informazioni. Tipo: Container	Sì

Nome	Description	Richiesto
Tier	Il livello di accesso ai dati da utilizzare per il ripristino dell'archivio. Standard è il valore predefinito. Tipo: Enum Valori validi: Expedited Standard Bulk Antenati: RestoreRequest	No
Description	La descrizione opzionale della richiesta. Tipo: stringa Antenati: RestoreRequest	No
SelectParameters	Descrive i parametri per la richiesta di selezione del lavoro. Tipo: Container Antenati: RestoreRequest	Sì, se il tipo di richiesta è SELECT
OutputLocation	Descrive la posizione che riceve i risultati della richiesta di ripristino di selezione. Tipo: Contenitore per Amazon S3 Antenati: RestoreRequest	Sì, se il tipo di richiesta è SELECT

L'elemento **SelectParameters** contenitore contiene i seguenti elementi.

Nome	Description	Richiesto
Expression	L'espressione SQL. Ad esempio:	Sì

Nome	Description	Richiesto
	L'espressione SQL seguente recupera la prima colonna dei dati dall'oggetto archiviato in formato CSV: <pre>SELECT s._1 FROM Object s</pre> La seguente espressione SQL restituisce tutto ciò che è contenuto nell'oggetto: <pre>SELECT * FROM Object</pre> Tipo: stringa Antenati: <code>SelectParameters</code>	
ExpressionType	Identifica il tipo di espressione. Tipo: stringa Valori validi: SQL Antenati: <code>SelectParameters</code>	Sì
InputSerialization	Descrive il formato di serializzazione dell'oggetto. Tipo: contenitore per CSV Antenati: <code>SelectParameters</code>	Sì
OutputSerialization	Descrive il modo in cui i risultati del processo Select sono serializzati. Tipo: contenitore per CSV Antenati: <code>SelectParameters</code>	Sì

L'elemento contenitore CSV nell'**InputSerialization** elemento contiene i seguenti elementi.

Nome	Description	Richiesto
RecordDelimiter	Un singolo carattere utilizzato per separare i singoli record nell'input. Invece del valore predefinito, è possibile specificare un delimitatore arbitrario. Tipo: stringa Impostazione predefinita: \n Antenati: CSV	No
FieldDelimiter	Un singolo carattere utilizzato per separare i singoli campi di un record. È possibile specificare un delimitatore arbitrario. Tipo: stringa Impostazione predefinita: , Antenati: CSV	No
QuoteCharacter	Un singolo carattere utilizzato per l'escape quando il delimitatore di campo fa parte del valore. Considera questo esempio in un file CSV: "a, b" Racchiudendo il valore tra virgolette, questo valore diventa un campo singolo. Se non si utilizzano le virgolette, la virgola funge da delimitatore di campo (il che lo rende due valori di campo separati, a e b). Tipo: stringa Impostazione predefinita: "	No

Nome	Description	Richiesto
	Antenati: CSV	
QuoteEscapeCharacter	Un singolo carattere utilizzato per l'escape delle virgolette all'interno di un valore già sfuggito. Ad esempio, il valore <code>"'"a , b '"'</code> viene analizzato come. <code>" a , b "</code> Tipo: stringa Impostazione predefinita: <code>"</code> Antenati: CSV	No
FileHeaderInfo	Descrive la prima riga dei dati di input. È uno dei valori ENUM. • NONE: La prima riga non è un'intestazione. • IGNORE: la prima riga è un'intestazione, ma non è possibile utilizzare i valori dell'intestazione per indicare la colonna in un'espressione. È possibile utilizzare la posizione della colonna (ad esempio <code>_1, _2,...</code>) per indicare la colonna (<code>()</code>). <code>SELECT s._1 FROM OBJECT s</code> • Use: la prima riga è un'intestazione ed è possibile utilizzare il valore dell'intestazione per identificare una colonna in un'espressione (<code>()</code>). <code>SELECT "name" FROM OBJECT</code> Tipo: Enum Valori validi: NONE USE IGNORE Antenati: CSV	No

Nome	Description	Richiesto
Comments	Un singolo carattere utilizzato per indicare che una riga deve essere ignorata quando il carattere è presente all'inizio della riga. È possibile specificare qualsiasi carattere per indicare una riga di commento. Tipo: stringa Antenati: CSV	No

L'elemento contenitore CSV (negli **OutputSerialization** elementi) contiene i seguenti elementi.

Nome	Description	Richiesto
QuoteFields	Indica se utilizzare le virgolette nei campi di output. • ALWAYS: utilizza sempre le virgolette per i campi di output. • ASNEEDED: utilizza le virgolette per i campi di output quando necessario. Tipo: Enum Valori validi: ALWAYS ASNEEDED Impostazione predefinita: AsNeeded Antenati: CSV	No
RecordDelimiter	Un singolo carattere utilizzato per separare i singoli record nell'output. Invece del valore predefinito, è possibile specificare un delimitatore arbitrario. Tipo: stringa	No

Nome	Description	Richiesto
	Impostazione predefinita: \n Antenati: CSV	
FieldDelimiter	Un singolo carattere utilizzato per separare i singoli campi di un record. È possibile specificare un delimitatore arbitrario. Tipo: stringa Impostazione predefinita: , Antenati: CSV	No
QuoteCharacter	Un singolo carattere utilizzato per l'escape quando il delimitatore di campo fa parte del valore. Ad esempio, se il valore è a , b, Amazon S3 racchiude il valore di questo campo tra virgolette, come segue: " a , b " Tipo: stringa Impostazione predefinita: " Antenati: CSV	No
QuoteEscapeCharacter	Un singolo carattere utilizzato per l'escape delle virgolette all'interno di un valore già sfuggito. Ad esempio, se il valore è " a , b " , Amazon S3 racchiude il valore tra virgolette, come segue: "" " a , b "" Tipo: stringa Antenati: CSV	No

L'elemento contenitore S3 (nell'**OutputLocation**elemento) contiene i seguenti elementi.

Nome	Description	Richiesto
AccessControlList	Un elenco di sovvenzioni che controllano l'accesso ai risultati suddivisi in fasi. Tipo: Container for Grant Antenati: S3	No
BucketName	Il nome del bucket S3 in cui sono archiviati i risultati del ripristino selezionato. Il bucket deve trovarsi nello stesso Regione AWS bucket che contiene l'oggetto di archivio di input. Tipo: stringa Antenati: S3	Sì
CannedACL	L'elenco di controllo degli accessi predefinito (ACL) da applicare ai risultati di ripristino selezionati. Tipo: stringa Valori validi: <code>private</code> <code>public-read</code> <code>public-read-write</code> <code>aws-exec-read</code> <code>authenticated-read</code> <code>bucket-owner-read</code> <code>bucket-owner-full-control</code> Antenati: S3	No
Encryption	Contiene informazioni di crittografia per i risultati archiviati. Tipo: contenitore per la crittografia Antenati: S3	No

Nome	Description	Richiesto
Prefix	Il prefisso che viene anteposto ai risultati del ripristino selezionato. La lunghezza massima del prefisso è 512 byte. Tipo: stringa Antenati: S3	Sì
StorageClass	La classe di archiviazione utilizzata per memorizzare i risultati della richiesta di selezione. Tipo: stringa Valori validi: STANDARD REDUCED_REDUNDANCY STANDARD_IA ONEZONE_IA Antenati: S3	No
Tagging	Contenitore per informazioni sui tag. Tipo: struttura dei tag Antenati: S3	No
UserMetadata	Contiene un elenco di metadati da archiviare con i risultati di ripristino selezionati. Tipo: struttura MetadataEntry Antenati: S3	No

L'elemento contenitore Grantee (nell'**AccessControlList** elemento) contiene i seguenti elementi.

Nome	Description	Richiesto
DisplayName	Il nome visualizzato dell'assegnatario.	No

Nome	Description	Richiesto
	Tipo: stringa Antenati: Grantee	
EmailAddress	L'indirizzo e-mail dell'assegnatario. Tipo: stringa Antenati: Grantee	No
ID	L'ID utente canonico ID dell'assegnatario. ■Tipo: stringa Antenati: Grantee	No
Type	Il tipo dell'assegnatario. Tipo: stringa Antenati: Grantee	No
URI	L'URI del gruppo di assegnatari. Tipo: stringa Antenati: Grantee	No
Permission	Autorizzazione concessa. Tipo: stringa Antenati: Grantee	No

L'elemento **contentione** Encryption (in S3) contiene i seguenti elementi.

Nome	Description	Richiesto
EncryptionType	L'algoritmo di crittografia lato server utilizzato per l'archiviazione dei risultati del lavoro. Il valore di default non prevede la crittografia. Tipo: stringa Valori validi <code>aws:kms</code> <code>AES256</code> Antenati: Encryption	No
KMSContext	Opzionale. Se il tipo di crittografia è <code>aws:kms</code>, è possibile utilizzare questo valore per specificare il contesto di crittografia per i risultati di ripristino selezionati. Tipo: stringa Antenati: crittografia	No
KMSKeyId	L'ID della chiave AWS Key Management Service (AWS KMS) da utilizzare per la crittografia degli oggetti. Tipo: stringa Antenati: Encryption	No

L'elemento **TagSet** contenitore (nell'elemento **Tagging**) **contiene il seguente** elemento.

Nome	Description	Richiesto
Tag	Contiene tag. Tipo: Container Antenati: TagSet	No

L'elemento contenitore **Tag** (nell'**TagSet** elemento) contiene i seguenti elementi.

Nome	Description	Richiesto
Key	Nome del tag. Tipo: stringa Antenati: Tag	No
Value	Valore del tag. Tipo: stringa Antenati: Tag	No

L'elemento **MetadataEntry** contenitore (nell'**UserMetadata** elemento) contiene i seguenti elementi di coppia chiave-valore da memorizzare con un oggetto.

Nome	Description	Richiesto
MetadataKey	La chiave di metadati. Tipo: stringa Antenati:	No
MetadataEntry	Il valore dei metadati. Tipo: stringa Antenati:	No

Risposte

Un'operazione riuscita restituisce il codice di **202 Accepted** stato **200 OK** o.

- Se la copia dell'oggetto non è stata ripristinata in precedenza, Amazon S3 restituisce 202 Accepted la risposta.
- Se la copia dell'oggetto è stata ripristinata in precedenza, Amazon S3 ritorna 200 OK nella risposta.

Intestazioni di risposta

Questa implementazione dell'operazione utilizza solo le intestazioni di risposta comuni alla maggior parte delle risposte. Per ulteriori informazioni, vedere [Common Response Headers](#).

Elementi di risposta

Questa operazione non restituisce elementi di risposta.

Errori speciali

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
RestoreAlreadyInProgress	Il ripristino degli oggetti è già in corso. (Questo errore non si applica alle richieste di tipo SELECT.)	409 Conflitto	Client
GlacierExpeditedRetrievalNotAvailable	I recuperi rapidi della classe di storage S3 Glacier non sono attualmente disponibili. Riprova più tardi. (Restituito se la capacità di elaborazione della richiesta è insufficiente. Expedited Questo errore si applica solo ai Expedited recuperi e non ai Standard Bulk recuperi.	503	N/A

Esempi

Ripristina un oggetto per due giorni utilizzando l'opzione di recupero rapido

La seguente richiesta di ripristino ripristina una copia dell'photo1.jpg oggetto da Amazon Glacier per un periodo di due giorni utilizzando l'opzione di recupero rapido.

```
POST /photo1.jpg?restore HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Mon, 22 Oct 2012 01:49:52 GMT
Authorization: authorization string
Content-Length: content length
```

```
<RestoreRequest>
  <Days>2</Days>
  <GlacierJobParameters>
    <Tier>Expedited</Tier>
  </GlacierJobParameters>
</RestoreRequest>
```

Se non examplebucket dispone di una copia ripristinata dell'oggetto, Amazon S3 restituisce la seguente risposta. 202 Accepted

```
HTTP/1.1 202 Accepted
x-amz-id-2: GFihv3y6+kE7KG11GEkQhU7/2/cHR3Yb2fCb2S04nxI423Dqwg2XiQ0B/
UZ1zYQvPiBlZNRcovw=
x-amz-request-id: 9F341CD3C4BA79E0
Date: Sat, 20 Oct 2012 23:54:05 GMT
Content-Length: 0
Server: AmazonS3
```

Se una copia dell'oggetto è già ripristinata, Amazon S3 restituisce una 200 OK risposta e aggiorna solo la data di scadenza della copia ripristinata.

Interroga un archivio con una richiesta SELECT

Di seguito è riportato un esempio di richiesta di ripristino di selezione.

```
POST /object-one.csv?restore HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Sat, 20 Oct 2012 23:54:05 GMT
Authorization: authorization string
```


Content-Length: *content length*

```
<RestoreRequest xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Type>SELECT</Type>
  <Tier>Expedited</Tier>
  <Description>this is a description</Description>
  <SelectParameters>
    <InputSerialization>
      <CSV>
        <FileHeaderInfo>IGNORE</FileHeaderInfo>
        <Comments>#</Comments>
        <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
        <RecordDelimiter>\n</RecordDelimiter>
        <FieldDelimiter>,</FieldDelimiter>
        <QuoteCharacter>"</QuoteCharacter>
      </CSV>
    </InputSerialization>
    <ExpressionType>SQL</ExpressionType>
    <Expression>select * from object</Expression>
    <OutputSerialization>
      <CSV>
        <QuoteFields>ALWAYS</QuoteFields>
        <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
        <RecordDelimiter>\n</RecordDelimiter>
        <FieldDelimiter>\t</FieldDelimiter>
        <QuoteCharacter>\'</QuoteCharacter>
      </CSV>
    </OutputSerialization>
  </SelectParameters>
  <OutputLocation>
    <S3>
      <BucketName>example-output-bucket</BucketName>
      <Prefix>test-s3</Prefix>
      <AccessControlList>
        <Grant>
          <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="AmazonCustomerByEmail">
            <EmailAddress>jane-doe@example.com</EmailAddress>
          </Grantee>
          <Permission>FULL_CONTROL</Permission>
        </Grant>
      </AccessControlList>
      <UserMetadata>
        <MetadataEntry>
```

```
<Name>test</Name>
<Value>test-value</Value>
</MetadataEntry>
<MetadataEntry>
  <Name>other</Name>
  <Value>something else</Value>
</MetadataEntry>
</UserMetadata>
<StorageClass>STANDARD</StorageClass>
</S3>
</OutputLocation>
</RestoreRequest>
```

Amazon S3 restituisce la seguente 202 Accepted risposta.

```
HTTP/1.1 202 Accepted
x-amz-id-2: GFihv3y6+kE7KG11GEkQhU7/2/cHR3Yb2fCb2S04nxI423Dqwg2XiQ0B/
UZ1zYQvPiB1ZNRcovw=
x-amz-request-id: 9F341CD3C4BA79E0
x-amz-restore-output-path: js-test-s3/qE8nk5M0XIj-LuZE2HXNw6empQm3znLkH1MWInRYPS-
0r12W0uj6LyYm-neTvm1-btz3wbBxfMhPykd3jkl-1vZE7w42/
Date: Sat, 20 Oct 2012 23:54:05 GMT
Content-Length: 0
Server: AmazonS3
```

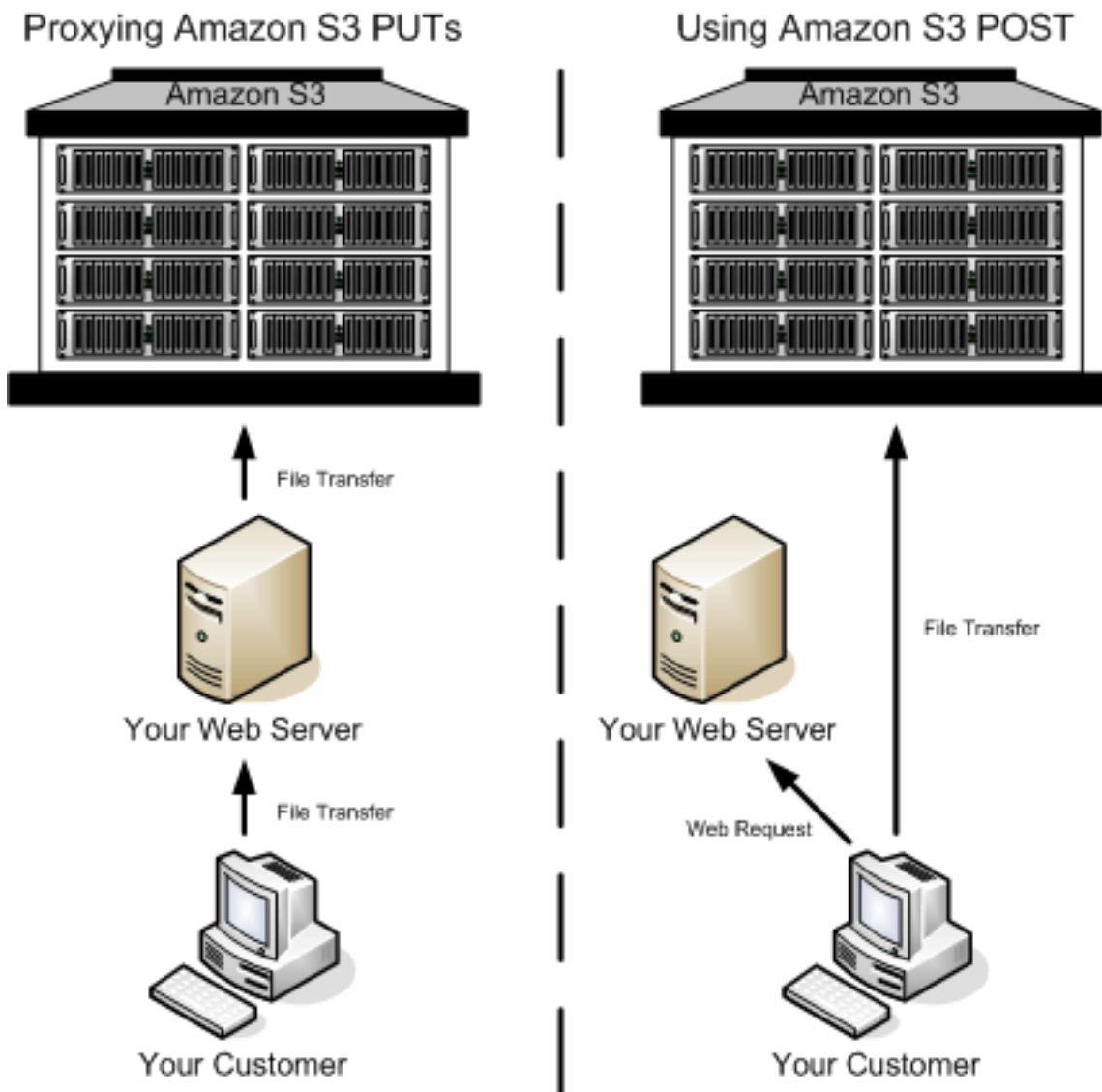
Ulteriori informazioni

- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)
- [SQL Reference per Amazon S3 Select e Amazon Glacier Select](#) nella Guida per l'utente di Amazon Simple Storage Service

Browser-Based Caricamenti tramite HTTP POST

Amazon S3 supporta le richieste HTTP POST in modo che gli utenti possano caricare contenuti direttamente su Amazon S3. Utilizzando POST, gli utenti finali possono autenticare le richieste senza dover passare i dati attraverso un nodo intermedio sicuro che protegge le tue credenziali. Pertanto, HTTP POST ha il potenziale per ridurre la latenza.

La figura seguente mostra un caricamento su Amazon S3 utilizzando una richiesta POST.



1. L'utente accede alla tua pagina da un browser web.
2. La tua pagina web contiene un modulo HTML che contiene tutte le informazioni necessarie all'utente per caricare contenuti su Amazon S3.
3. L'utente carica i contenuti su Amazon S3 tramite il browser web.

La procedura per l'invio di richieste POST basate su browser è la seguente:

1. Crea una politica di sicurezza che specifichi le condizioni che limitano ciò che desideri consentire nella richiesta, ad esempio il nome del bucket in cui è possibile caricare gli oggetti e i prefissi dei nomi delle chiavi che desideri consentire per l'oggetto che viene creato.

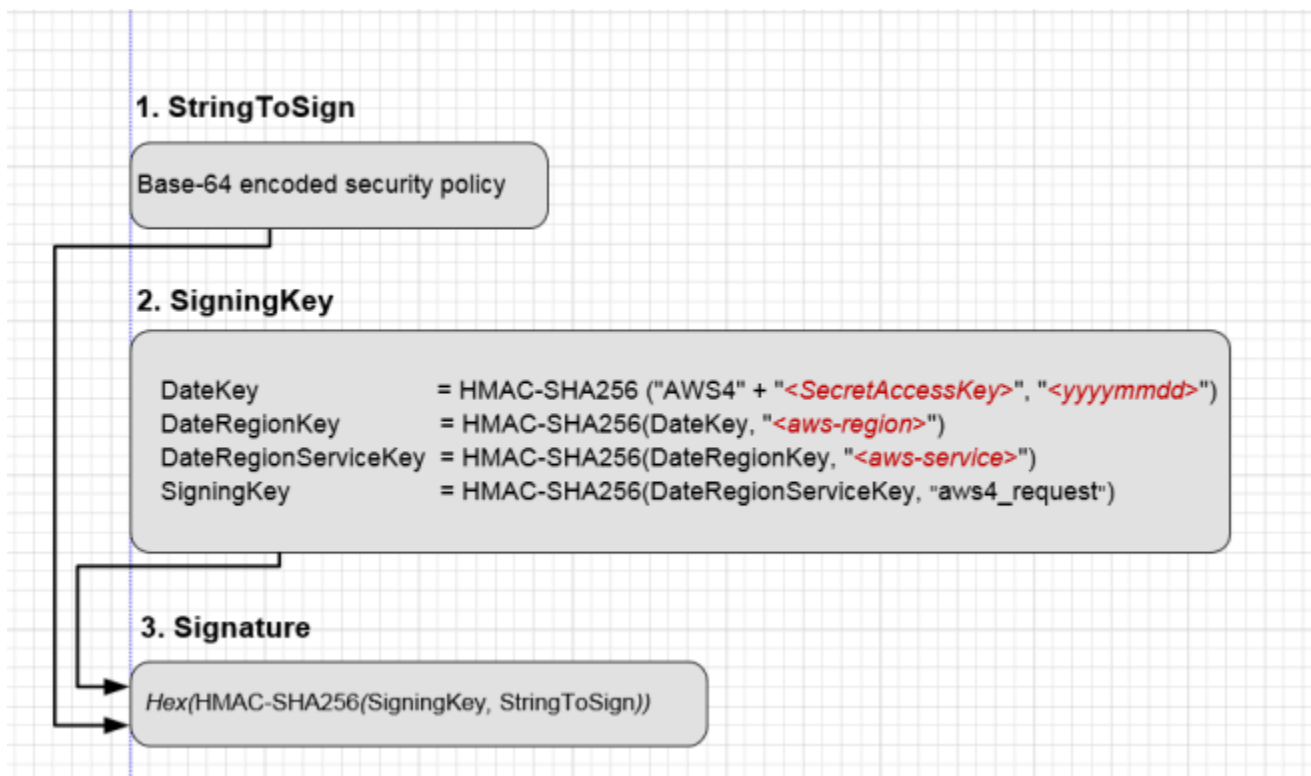
2. Crea una firma basata sulla policy. Per le richieste autenticate, il modulo deve includere una firma valida e la politica.
3. Crea un modulo HTML a cui i tuoi utenti possano accedere per caricare oggetti nel tuo bucket Amazon S3.

La sezione seguente descrive come creare una firma per autenticare una richiesta. Per informazioni sulla creazione di moduli e politiche di sicurezza, vedere [Creazione di un modulo HTML \(utilizzando AWS Signature \(versione 4\)\)](#).

Calcolo di una firma

Per le richieste autenticate, il modulo HTML deve includere i campi per una politica di sicurezza e una firma.

- Una politica di sicurezza (vedi [Politica POST](#)) controlla ciò che è consentito nella richiesta.
- La politica di sicurezza è la StringToSign (vedi [Introduzione alla firma delle richieste](#)) nel calcolo della firma.



Per calcolare una firma

1. Crea una politica utilizzando la UTF-8 codifica.
2. Converti i byte UTF-8-encoded della policy in base64. Il risultato è il `StringToSign`
3. Crea una chiave di firma.
4. Usa la chiave di firma per firmare l'algoritmo di HMAC-SHA256 firma in `StringToSign` uso.

Per ulteriori informazioni sulla creazione di moduli HTML, politiche di sicurezza e un esempio, vedere quanto segue:

- [Creazione di un modulo HTML \(utilizzando AWS Signature \(versione 4\)\)](#)
- [Politica POST](#)
- [Esempio: Browser-Based caricamento tramite HTTP POST \(utilizzando AWS Signature \(versione 4\)\)](#)

Creazione di un modulo HTML (utilizzando AWS Signature (versione 4))

Argomenti

- [Dichiarazione del modulo HTML](#)
- [Campi del modulo HTML](#)

Per consentire agli utenti di caricare contenuti su Amazon S3 utilizzando i loro browser (richieste HTTP POST), utilizza i moduli HTML. I moduli HTML si compongono di una dichiarazione del modulo e di campi del modulo. La dichiarazione del modulo contiene informazioni generali sulla richiesta. I campi del modulo contengono informazioni dettagliate sulla richiesta.

Questa sezione descrive come creare moduli HTML. Per un esempio funzionante di caricamento basato su browser tramite HTTP POST e relativi calcoli delle firme per l'autenticazione delle richieste, vedere [Esempio: Browser-Based caricamento tramite HTTP POST \(utilizzando AWS Signature \(versione 4\)\)](#)

Il modulo e la policy devono essere UTF-8 codificati. È possibile applicare la UTF-8 codifica al modulo specificando `charset=UTF-8` nell'attributo `content`. Di seguito è riportato un esempio di UTF-8 codifica nell'intestazione HTML.

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

Di seguito è riportato un esempio di UTF-8 codifica nell'intestazione di una richiesta.

```
Content-Type: text/html; charset=UTF-8
```

Note

I dati e i limiti del modulo (escluso il contenuto del file) non possono superare i 20 KB.

Dichiarazione del modulo HTML

La dichiarazione del modulo HTML ha i tre attributi seguenti:

- **action**— L'URL che elabora la richiesta, che deve essere impostato sull'URL del bucket. Ad esempio, se il nome del bucket è `examplebucket`, l'URL è `http://examplebucket.s3.amazonaws.com/`

Note

Il nome della chiave è specificato nel campo del modulo.

- **method**— Il metodo deve essere POST.
- **enctype**— Il tipo di allegato (`enctype`) deve essere impostato su `multipart/form-data` sia per i caricamenti di file che per i caricamenti in aree di testo. Per ulteriori informazioni `enctype`, vedere RFC 1867. <http://www.ietf.org/rfc/rfc1867.txt>

Questo è un modulo di dichiarazione per il bucket. `examplebucket`

```
<form action="http://examplebucket.s3.amazonaws.com/" method="post"
  enctype="multipart/form-data">
```

Campi del modulo HTML

La tabella seguente descrive un elenco di campi che è possibile utilizzare all'interno di un modulo. Tra gli altri campi, c'è un campo per la firma che puoi usare per autenticare le richieste. Sono disponibili campi per specificare l'algoritmo di calcolo della firma (`x-amz-algorithm`), l'ambito delle credenziali (`x-amz-credential`) utilizzato per generare la chiave di firma e la data (`x-amz-date`) utilizzata per calcolare la firma. Amazon S3 utilizza queste informazioni per ricreare la firma. Se le firme corrispondono, Amazon S3 elabora la richiesta.

Note

La variabile `${filename}` viene sostituita automaticamente con il nome del file fornito dall'utente ed è riconosciuta da tutti i campi del modulo. Se il browser o il client fornisce un percorso completo o parziale al file, viene utilizzato solo il testo che segue l'ultima barra (/) o la barra rovesciata (\) (ad esempio, `C:\Program Files\directory1\file.txt` viene interpretato come). `file.txt` Se non vengono forniti file o nomi di file, la variabile viene sostituita con una stringa vuota.

Se non fornisci gli elementi necessari per le richieste autenticate, come l'`policyelemento`, si presume che la richiesta sia anonima e avrà esito positivo solo se hai configurato il bucket per la lettura e la scrittura pubbliche.

Nome elemento	Description	Richiesto
<code>acl</code>	Lista di controllo accessi (ACL) Amazon S3. Se viene specificato un ACL non valido, Amazon S3 rifiuta la richiesta. Per ulteriori informazioni sugli ACL, consulta Utilizzo degli ACL di Amazon S3 . Tipo: stringa	No

Nome elemento	Description	Richiesto
	Default: private Valori validi: private public-read public-read-write aws-exec-read authenticated-read bucket-owner-read bucket-owner-full-control	
Cache-Control	REST-specific intestazioni. Per ulteriori informazioni, consulta PutObject .	No
Content-Type		
Content-Disposition		
Content-Encoding		
Expires		
key	Il nome della chiave dell'oggetto caricato. Per utilizzare il nome di file fornito dall'utente, usa la variabile <code>\${filename}</code>. Ad esempio, se carichi un file <code>photo1.jpg</code> e lo specifichi <code>/user/user1/\${filename}</code> come nome chiave, il file viene archiviato come <code>/user/user1/photo1.jpg</code> Per ulteriori informazioni, consulta Object Key and Metadata nella Amazon Simple Storage Service User Guide.	Sì

Nome elemento	Description	Richiesto
<code>policy</code>	La politica di sicurezza con codifica base64 che descrive cosa è consentito nella richiesta . Per le richieste autenticate, è richiesta una policy. Le richieste senza una politica di sicurezza sono considerate anonime e avranno esito positivo solo su un bucket scrivibile pubblicamente.	Richiesto per le richieste autenticate
<code>success_action_redirect</code>	L'URL a cui viene reindirizzato il client dopo il corretto caricamento. Se non <code>success_action_redirect</code> è specificato o Amazon S3 non è in grado di interpretare l'URL, Amazon S3 restituisce il tipo di documento vuoto specificato nel campo. <code>success_action_status</code> Se il caricamento non riesce, Amazon S3 restituisce un errore e non reindirizza l'utente a un altro URL.	No

Nome elemento	Description	Richiesto
success_action_status	Il codice di stato restituito al client in caso di caricamento riuscito, se non <code>success_action_redirect</code> è specificato. I valori validi sono 200, 201, o 204 (impostazione predefinita). Se il valore è impostato su 200 o 204, Amazon S3 restituisce un documento vuoto con il codice di stato specificato. Se il valore è impostato su 201, Amazon S3 restituisce un documento XML con codice di stato 201. Per informazioni sul contenuto del documento XML, consulta POST Object. Se il valore non è impostato o non è valido, Amazon S3 restituisce un documento vuoto con un codice di stato 204.  Note Alcune versioni di Adobe Flash Player non gestiscono correttamente le risposte HTTP con corpo vuoto. Per supportare i caricamenti tramite Adobe Flash, si consiglia di impostare <code>success_action_status</code> su 201.	No

Nome elemento	Description	Richiesto
x-amz-algorithm	L'algoritmo di firma utilizzato per autenticare la richiesta. Per AWS Signature Version 4, il valore è AWS4-HMAC-SHA256 . Questo campo è obbligatorio se la richiesta include un documento di policy.	Obbligatorio per le richieste autenticate
x-amz-credential	Oltre all'ID della chiave di accesso, questo campo fornisce anche informazioni sull'ambito che identificano la regione e il servizio per i quali la firma è valida. Questo dovrebbe essere lo stesso ambito utilizzato nel calcolo della chiave di firma per il calcolo della firma. È una stringa del seguente formato: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Ad esempio: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</pre> Per Amazon S3, la stringa <code>aws-service</code> è <code>s3</code>. Per un elenco delle <code>aws-region</code> stringhe Amazon S3, consulta Regioni ed endpoint in. Riferimenti generali di AWS Ciò è necessario o se nella richiesta è incluso un documento di policy.	Obbligatorio per le richieste autenticate

Nome elemento	Description	Richiesto
x-amz-date	È il valore della data in formato ISO8601. Ad esempio, 20130728T000000Z . È la stessa data utilizzata per creare la chiave di firma (ad esempio, 20130728). Inoltre, deve essere lo stesso valore fornito nella polizza (x-amz-date) che hai firmato. Questo è necessario se nella richiesta è incluso un documento di polizza.	Obbligatorio per le richieste autenticate
x-amz-security-token	Un token di sicurezza utilizzato da Amazon DevPay e le credenziali di sessione Se la richiesta utilizza Amazon DevPay, richiede due campi del x-amz-security-token modulo: uno per il token del prodotto e uno per il token utente. Per ulteriori informazioni, consulta Using DevPay in the Amazon Simple Storage Service User Guide. Se la richiesta utilizza le credenziali di sessione, richiede un x-amz-security-token modulo. Per ulteriori informazioni, consulta Richiesta di credenziali di sicurezza temporanee nella Guida per l'utente di IAM.	No
x-amz-signature	(AWS Signature Version 4) L' HMAC-SHA256 hash della politica di sicurezza. Questo campo è obbligatorio se la richiesta include un documento di policy.	Richiesto per le richieste autenticate

Nome elemento	Description	Richiesto
x-amz-meta-*	I nomi dei campi che iniziano con questo prefisso sono metadati definiti dall'utente. Ciascuno di essi viene archiviato e restituito come set di coppie chiave-valore. Amazon S3 non convalida né interpreta i metadati definiti dall'utente. Per ulteriori informazioni, consulta PutObject .	No
x-amz-*	Vedi POST Object () per altre intestazioni. POST Object x-amz-*	No
file	File o contenuto di testo. Il file o il contenuto deve essere l'ultimo campo del modulo. Non è possibile caricare più di un file alla volta.	Sì

Gli elementi condizionali sono obbligatori per le richieste autenticate e sono opzionali per le richieste anonime.

Ora che sai come creare moduli, puoi creare una politica di sicurezza che puoi firmare. Per ulteriori informazioni, consulta [Politica POST](#).

Politica POST

Argomenti

- [Scadenza](#)
- [Corrispondenza delle condizioni](#)
- [Condizioni](#)
- [Utilizzo di escape con caratteri](#)

La politica richiesta per effettuare richieste autenticate tramite HTTP POST è un UTF-8 documento con codifica base64 scritto in JavaScript Object Notation (JSON) che specifica le condizioni che la richiesta deve soddisfare. A seconda di come si progetta il documento di policy, è possibile controllare la granularità dell'accesso per caricamento, per utente, per tutti i caricamenti o in base ad altri design che soddisfano le proprie esigenze.

Questa sezione descrive la politica POST. Ad esempio, i calcoli delle firme utilizzando la politica POST, vedere [Esempio: Browser-Based caricamento tramite HTTP POST \(utilizzando AWS Signature versione 4\)](#).

Note

Sebbene il documento sulla politica sia facoltativo, ti consigliamo vivamente di utilizzarne uno per controllare cosa è consentito nella richiesta. Se rendi il bucket scrivibile pubblicamente, non hai alcun controllo su quali utenti possono scrivere nel tuo bucket.

Di seguito è riportato un esempio di documento di policy POST.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"acl": "public-read" },
    {"bucket": "johnsmith" },
    ["starts-with", "$key", "user/eric/"],
  ]
}
```

La politica POST contiene sempre `conditions` gli elementi `expiration` and. La politica di esempio utilizza due tipi di corrispondenza delle condizioni (corrispondenza esatta e corrispondenza iniziale). Le sezioni seguenti descrivono questi elementi.

Scadenza

L'expirationelemento specifica la data e l'ora di scadenza della politica POST nel formato di data GMT ISO8601. Ad esempio, `2013-08-01T12:00:00.000Z` specifica che la politica POST non è valida dopo la mezzanotte GMT del 1° agosto 2013.

Corrispondenza delle condizioni

Di seguito è riportata una tabella che descrive i tipi di corrispondenza delle condizioni che è possibile utilizzare per specificare le condizioni della politica POST (descritti nella sezione successiva).

Sebbene sia necessario specificare almeno una condizione per ogni campo del modulo specificato nel modulo, è possibile creare criteri di corrispondenza più complessi specificando più condizioni per un campo modulo.

Tipo di corrispondenza delle condizioni	Description
Corrispondenze esatte	Il valore del campo del modulo deve corrispondere al valore specificato. Questo esempio indica che l'ACL deve essere impostata su public-read: <pre>{"acl": "public-read" }</pre> Questo esempio rappresenta un modo alternativo per indicare che l'ACL deve essere impostata su public-read: <pre>["eq", "\$acl", "public-read"]</pre>
Inizia con	Il valore deve iniziare con il valore specificato. Questo esempio indica che la chiave dell'oggetto deve iniziare con user/user1: <pre>["starts-with", "\$key", "user/user1/"]</pre>
Content-Type Corrispondenza in un Comma-Separated elenco	Content-Types i valori per una starts-with condizione che includono virgole vengono interpretati come elenchi. Ogni valore nell'elenco deve soddisfare la condizione affinché l'intera condizione venga soddisfatta. Ad esempio, se si verifica la seguente condizione: <pre>["starts-with", "\$Content-Type", "image/"]</pre>

Tipo di corrispondenza delle condizioni	Description
	Il valore seguente soddisferebbe la condizione: <pre>"image/jpg,image/png,image/gif"</pre> Il valore seguente non soddisferebbe la condizione: <pre>["image/jpg,text/plain"]</pre>  Note Gli elementi di dati diversi da quelli Content-Type vengono trattati come stringhe, indipendentemente dalla presenza di virgole.
Corrispondenza di qualsiasi contenuto	Per configurare la politica POST in modo da consentire qualsiasi contenuto all'interno di un campo del modulo, utilizzala <code>starts-with</code> con un valore vuoto («»). Questo esempio consente qualsiasi valore per <code>success_action_redirect</code> : <pre>["starts-with", "\$success_action_redirect", ""]</pre>
Specifica degli intervalli	Per i campi del modulo che accettano un intervallo, separa il limite superiore e inferiore con una virgola. Questo esempio consente una dimensione del file compresa tra 1 e 10 MiB: <pre>["content-length-range", 1048576, 10485760]</pre>

Le condizioni specifiche supportate in una politica POST sono descritte in [Condizioni](#).

Condizioni

`conditions` In una policy POST c'è una serie di oggetti, ognuno dei quali viene utilizzato per convalidare la richiesta. È possibile utilizzare queste condizioni per limitare ciò che è consentito nella richiesta. Ad esempio, le condizioni politiche precedenti richiedono quanto segue:

- La richiesta deve specificare il nome del `johnsmith` bucket.
- Il nome della chiave dell'oggetto deve avere il `user/eric` prefisso.
- L'ACL dell'oggetto deve essere impostato su `public-read`

Ogni campo del modulo specificato in un modulo (ad eccezione `dix-amz-signature`, `filepolicy`, e i nomi di campo con un `x-ignore-` prefisso) deve apparire nell'elenco delle condizioni.

Note

Tutte le variabili all'interno del modulo vengono espansive prima della convalida della politica POST. Pertanto, tutte le condizioni che corrispondono devono corrispondere ai campi del modulo espanso. Supponiamo di voler limitare il nome della chiave dell'oggetto a un prefisso specifico (`user/user1`). In questo caso, impostate il campo del modulo chiave su `user/user1/${filename}`. La tua politica POST dovrebbe essere `[ "starts-with", "$key", "user/user1/" ]` (non inserire `[ "starts-with", "$key", "user/user1/${filename}" ]`). Per ulteriori informazioni, consulta [Corrispondenza delle condizioni](#).

Le condizioni del documento politico sono descritte nella tabella seguente.

Nome elemento	Description
<code>acl</code>	Specifica il valore ACL che deve essere utilizzato nell'invio del modulo. Questa condizione supporta la corrispondenza esatta e il tipo di corrispondenza <code>starts-with</code> delle condizioni descritti nella sezione seguente.

Nome elemento	Description
bucket	Specifica il nome del bucket accettabile. Questa condizione supporta il tipo di corrispondenza esatta della condizione.
content-length-range	Le dimensioni minime e massime consentite per il contenuto caricato. Questa condizione supporta il tipo di corrispondenza content-length-range delle condizioni.
Cache-Control Content-Type Content-Disposition Content-Encoding Expires	REST-specific intestazioni. Per ulteriori informazioni, consulta POST Object. Questa condizione supporta la corrispondenza esatta e il tipo di corrispondenza starts-with delle condizioni.
key	Il nome della chiave accettabile o il prefisso dell'oggetto caricato. Questa condizione supporta la corrispondenza esatta e il tipo di corrispondenza starts-with delle condizioni.
success_action_redirect redirect	L'URL a cui viene reindirizzato il client dopo il corretto caricamento. Questa condizione supporta la corrispondenza esatta e il tipo di corrispondenza starts-with delle condizioni.

Nome elemento	Description
success_action_status	Il codice di stato restituito al client in caso di caricamento riuscito, se non success_action_redirect è specificato. Questa condizione supporta una corrispondenza esatta.
x-amz-algorithm	L'algoritmo di firma che deve essere utilizzato durante il calcolo della firma. Per la versione 4 di AWS Signature, il valore è AWS4-HMAC-SHA256 . Questa condizione supporta una corrispondenza esatta.
x-amz-credential	Le credenziali utilizzate per calcolare la firma. Fornisce l'ID della chiave di accesso e le informazioni sull'ambito che identificano l'area e il servizio per i quali la firma è valida. Dovrebbe essere lo stesso ambito utilizzato nel calcolo della chiave di firma per il calcolo della firma. È una stringa del seguente formato: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Ad esempio: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</pre> Per Amazon S3, la stringa aws-service è. s3 Per un elenco delle aws-region stringhe Amazon S3, consulta Regioni ed endpoint in. Riferimenti generali di AWS Ciò è necessario o se nella richiesta è incluso un documento di policy POST. Questa condizione supporta una corrispondenza esatta.

Nome elemento	Description
x-amz-date	Il valore della data specificato nella stringa in formato ISO8601. Ad esempio, 20130728T000000Z . La data deve essere la stessa utilizzata per creare la chiave di firma per il calcolo della firma. Ciò è necessario se nella richiesta è incluso un documento relativo alla politica POST. Questa condizione supporta una corrispondenza esatta.
x-amz-security-token	Token DevPay di sicurezza Amazon. Ogni richiesta che utilizza Amazon DevPay richiede due campi del x-amz-security-token modulo: uno per il token del prodotto e uno per il token utente. Di conseguenza, i valori devono essere separati da virgole. Ad esempio, se il token utente è eW91dHVIZQ== e il token del prodotto èb0hnNVNKWVJIQTA= , imposti la voce della politica POST su:{ "x-amz-security-token": "eW91dHVIZQ==,b0hnNVNKWVJIQTA=" } . Per ulteriori informazioni su Amazon DevPay, consulta Using DevPay in the Amazon Simple Storage Service User Guide.
x-amz-meta-*	User-specified metadati. Questa condizione supporta la corrispondenza esatta e il tipo di corrispondenza starts-with delle condizioni.
x-amz-*	Vedi POST Object () POST Object per altre x-amz- * intestazioni. Questa condizione supporta una corrispondenza esatta.

Note

Se il tuo toolkit aggiunge altri campi modulo (ad esempio, Flash aggiunge `filename`), devi aggiungerli al documento sulla politica POST. Se puoi controllare questa funzionalità, aggiungi il prefisso `x-ignore-` al campo in modo che Amazon S3 ignori la funzionalità e non ne alteri le versioni future.

Utilizzo di escape con caratteri

I caratteri che devono essere evitati all'interno di un documento di policy POST sono descritti nella tabella seguente.

Sequenza di escape	Descrizione
<code>\\</code>	Barra rovesciata
<code>\\$</code>	Simbolo del dollaro
<code>\b</code>	Backspace
<code>\f</code>	Avanzamento modulo
<code>\n</code>	Nuova riga
<code>\r</code>	Ritorno a capo
<code>\t</code>	Tabulatore orizzontale
<code>\v</code>	Tabulatore verticale
<code>\u xxxx</code>	Tutti i caratteri Unicode

Ora che conosci i moduli e le politiche e hai capito come funziona la firma, puoi provare un esempio di caricamento POST. È necessario scrivere il codice per calcolare la firma. L'esempio fornisce un modulo di esempio e una politica POST che puoi utilizzare per testare i calcoli della firma. Per ulteriori informazioni, consulta [Esempio: Browser-Based caricamento tramite HTTP POST \(utilizzando AWS Signature \(versione 4\)\)](#).

Esempio: Browser-Based caricamento tramite HTTP POST (utilizzando AWS Signature (versione 4))

Questa sezione mostra un esempio di utilizzo di una richiesta HTTP POST per caricare contenuti direttamente su Amazon S3.

Per ulteriori informazioni sulla versione 4 di Signature, consulta [Signature Version 4 Signing Process](#).

Caricamento di un file su Amazon S3 tramite HTTP POST

Questo esempio fornisce un esempio di policy POST e un modulo che puoi utilizzare per caricare un file. L'argomento utilizza il criterio di esempio e le credenziali fittizie per mostrare il flusso di lavoro e l'hash della firma e della politica risultanti. Puoi utilizzare questi dati come suite di test per verificare il codice di calcolo della firma.

L'esempio utilizza le seguenti credenziali di esempio per il calcolo della firma. È possibile utilizzare queste credenziali per verificare il codice di calcolo della firma. Tuttavia, è necessario sostituirle con le proprie credenziali quando si inviano richieste a. AWS

Parametro	Valore
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

Esempio di politica e modulo

La seguente politica POST supporta i caricamenti su Amazon S3 con condizioni specifiche.

```
{ "expiration": "2015-12-30T12:00:00.000Z",
  "conditions": [
    {"bucket": "sigv4examplebucket"},
    ["starts-with", "$key", "user/user1/"],
    {"acl": "public-read"},
    {"success_action_redirect": "http://sigv4examplebucket.s3.amazonaws.com/successful_upload.html"},
    ["starts-with", "$Content-Type", "image/"],
    {"x-amz-meta-uuid": "14365123651274"}]
```

```
{
  "x-amz-server-side-encryption": "AES256",
  "starts-with": "$x-amz-meta-tag", ""],

  "x-amz-credential": "AKIAIOSFODNN7EXAMPLE/20151229/us-east-1/s3/aws4_request",
  "x-amz-algorithm": "AWS4-HMAC-SHA256",
  "x-amz-date": "20151229T000000Z" }
]
```

Questa politica POST stabilisce le seguenti condizioni sulla richiesta:

- Il caricamento deve avvenire entro mezzogiorno UTC del 30 dicembre 2015.
- Il contenuto può essere caricato solo su. `sigv4examplebucket` Il bucket deve trovarsi nella regione specificata nell'ambito delle credenziali (parametro del `x-amz-credential` modulo), poiché la firma fornita è valida solo all'interno di questo ambito.
- Puoi fornire qualsiasi nome di chiave che inizi con. `user/user1` Ad esempio, `user/user1/MyPhoto.jpg`.
- L'ACL deve essere impostato su. `public-read`
- Se il caricamento ha esito positivo, il browser dell'utente viene reindirizzato a. `http://sigv4examplebucket.s3.amazonaws.com/successful_upload.html`
- L'oggetto deve essere un file di immagine.
- Il `x-amz-meta-uuid` tag deve essere impostato su `14365123651274`.
- `x-amz-meta-tag` Può contenere qualsiasi valore.

Quella che segue è una Base64-encoded version di questa politica POST. Questo valore viene utilizzato come calcolo `StringToSign` della firma.

```
eyJhZiZxhwaXJhdGlvbiI6ICIyMDE1LTUyLTMwVDEyOjAwOjAwLjAwMFoiLA0KICAiY29uZGl0aW9ucyI6IFsNCiAgICB7ImJ
```

Quando applichi copy/paste la politica precedente, dovrebbe contenere ritorni a capo e nuove righe affinché l'hash calcolato corrisponda a questo valore (ad es. Testo ASCII, con terminatori di riga CRLF).

Utilizzando le credenziali di esempio per creare una firma, il valore della firma è il seguente (nel calcolo della firma, la data è la stessa indicata `x-amz-date` nella policy (20151229):

```
8afdbf4008c03f22c2cd3cdb72e4afbb1f6a588f3255ac628749a66d7f09699e
```


Il seguente modulo di esempio specifica la precedente politica POST e supporta una richiesta POST a. `sigv4examplebucket` Copy/paste il contenuto in un editor di testo e salvalo come `exampleform.html`. È quindi possibile caricare i file di immagine nel bucket specifico utilizzando il file `exampleform.html`. La tua richiesta avrà esito positivo se la firma fornita corrisponde alla firma calcolata da Amazon S3.

Note

È necessario aggiornare il nome del bucket, le date, le credenziali, la policy e la firma con valori validi affinché il bucket venga caricato correttamente su S3.

```
<html>
  <head>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

  </head>
  <body>

    <form action="http://sigv4examplebucket.s3.amazonaws.com/" method="post"
    enctype="multipart/form-data">
      Key to upload:
      <input type="input" name="key" value="user/user1/${filename}" /><br />
      <input type="hidden" name="acl" value="public-read" />
      <input type="hidden" name="success_action_redirect" value="http://
sigv4examplebucket.s3.amazonaws.com/successful_upload.html" />
      Content-Type:
      <input type="input" name="Content-Type" value="image/jpeg" /><br />
      <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
      <input type="hidden" name="x-amz-server-side-encryption" value="AES256" />
      <input type="text" name="X-Amz-Credential" value="AKIAIOSFODNN7EXAMPLE/20151229/
us-east-1/s3/aws4_request" />
      <input type="text" name="X-Amz-Algorithm" value="AWS4-HMAC-SHA256" />
      <input type="text" name="X-Amz-Date" value="20151229T000000Z" />

      Tags for File:
      <input type="input" name="x-amz-meta-tag" value="" /><br />
      <input type="hidden" name="Policy" value='<Base64-encoded policy string>' />
      <input type="hidden" name="X-Amz-Signature" value="<signature-value>" />
      File:
      <input type="file" name="file" /> <br />
```

```
<!-- The elements after this will be ignored -->
<input type="submit" name="submit" value="Upload to Amazon S3" />
</form>

</html>
```

I parametri del post non fanno distinzione tra maiuscole e minuscole. Ad esempio, puoi specificare `x-amz-signature` o `X-Amz-Signature`.

Browser-based vengono caricati su Amazon S3 utilizzando AWS Libreria Amplify

Questa sezione descrive come caricare file su Amazon S3 utilizzando la libreria Amplify. AWS JavaScript

Per informazioni sulla configurazione della libreria AWS Amplify, consulta [Installazione e configurazione di AWS Amplify](#).

Utilizzo di AWS JavaScript Libreria Amplify per caricare file su Amazon S3

Il Storage modulo della libreria AWS Amplify offre un semplice meccanismo di caricamento basato su browser per la gestione dei contenuti degli utenti nello storage pubblico o privato di Amazon S3.

Example: AWS Configurazione manuale di Amplify

L'esempio seguente mostra la configurazione manuale per l'utilizzo del modulo AWS Amplify Storage. L'implementazione predefinita del modulo Storage utilizza Amazon S3.

```
import Amplify from 'aws-amplify';
Amplify.configure({
  Auth: {
    identityPoolId: 'XX-XXXX-X:XXXXXXXXXX-XXXX-1234-abcd-1234567890ab', //REQUIRED -
    Amazon Cognito Identity Pool ID
    region: 'XX-XXXX-X', // REQUIRED - Amazon Cognito Region
    userPoolId: 'XX-XXXX-X_abcd1234', //OPTIONAL - Amazon Cognito User Pool ID
    userPoolWebClientId: 'XX-XXXX-X_abcd1234', //OPTIONAL - Amazon Cognito Web
    Client ID
  },
  Storage: {
    bucket: '', //REQUIRED - Amazon S3 bucket
```

```
    region: 'XX-XXXX-X', //OPTIONAL - Amazon service region
  }
);
```

Example: Inserisci i dati in Amazon S3

L'esempio seguente mostra come rendere put pubblici i dati in Amazon S3.

```
Storage.put('test.txt', 'Hello')
  .then (result => console.log(result))
  .catch(err => console.log(err));
```

L'esempio seguente mostra come rendere put privati i dati in Amazon S3.

```
Storage.put('test.txt', 'Private Content', {
  level: 'private',
  contentType: 'text/plain'
})
  .then (result => console.log(result))
  .catch(err => console.log(err));
```

Per ulteriori informazioni sull'utilizzo del modulo AWS Amplify Storage, consulta [AWS Amplify Storage](#).

Ulteriori informazioni

[AWS Avvio rapido di Amplify](#)

Intestazioni di richiesta comuni

La tabella seguente descrive le intestazioni che possono essere utilizzate da vari tipi di richieste REST di Amazon S3.

Header Name (Nome intestazione)	Description
Authorization	Le informazioni necessarie per l'autenticazione della richiesta. Per ulteriori informazioni, consulta The Authentication Header nella Amazon Simple Storage Service Developer Guide. Per le richieste anonime questa intestazione non è richiesta.
Access-Control-Request-Method	Un elenco di metodi HTTP inviato come richiesta CORS pre-flight. Se la valutazione CORS precedente al flight ha esito positivo, è consentito utilizzare i metodi specificati nella seguente richiesta CORS.
Content-Length	La lunghezza del messaggio (senza le intestazioni) secondo la specifica RFC 2616. Questa intestazione è necessaria per i PUT e le operazioni che caricano XML, come la registrazione e gli ACL.
Content-Type	Il tipo di contenuto della risorsa nel caso in cui la richiesta contenga contenuti nel corpo. Ad esempio: text/plain
Content-MD5	Il digest MD5 a 128 bit codificato in base64 del messaggio (senza le intestazioni) secondo RFC 1864. Questa intestazione può essere utilizzata come controllo dell'integrità del messaggio per verificare che i dati siano gli stessi dati originariamente inviati. Sebbene sia facoltativo, consigliamo di utilizzare il Content-MD5 meccanismo come controllo di integrità end-to-end. Per ulteriori informazioni sull'autenticazione delle richieste REST, consulta la sezione Autenticazione REST nella Amazon Simple Storage Service Developer Guide.

Header Name (Nome intestazione)	Description
Date	La data che può essere utilizzata per creare la firma contenuta nell'intestazione <code>Authorization</code>. Se per la firma è necessario utilizzare l'intestazione <code>Date</code>, questa deve essere specificata nel formato di base ISO 8601. In questo caso, l'intestazione <code>x-amz-date</code> non è necessaria. Nota che quando <code>x-amz-date</code> è presente, sovrascrive sempre il valore dell'intestazione <code>Date</code>. Se l'intestazione <code>Date</code> non viene utilizzata per la registrazione, può essere uno dei formati di data completa specificati dal protocollo RFC 2616, sezione 3.3. Ad esempio, <code>date/time Wed, 01 Mar 2006 12:00:00 GMT</code> è un' <code>date/time</code> intestazione valida da utilizzare con Amazon S3. Se utilizzi l'intestazione <code>Date</code> per la firma, deve essere nel formato di base ISO 8601 <code>YYYYMMDD'T'HHMMSS'Z'</code>. Se <code>Date</code> è specificata ma non è nel formato base ISO 8601, devi includere anche l'intestazione <code>x-amz-date</code>. Se <code>Date</code> è nel formato di base ISO 8601, è sufficiente per firmare le richieste e l'intestazione <code>x-amz-date</code> non è necessaria. Per ulteriori informazioni, consulta Gestione delle date in Signature Version 4 nel glossario di Amazon Web Services.
Expect	Quando l'applicazione utilizza <code>100-continue</code>, non invia il corpo della richiesta finché non riceve una conferma. Se, in base alle intestazioni, il messaggio viene rifiutato, il corpo del messaggio non viene inviato. Questa intestazione può essere utilizzata solo se si invia un corpo. Valori validi: <code>100-continue</code>

Header Name (Nome intestazione)	Description
Host	Per le richieste in stile percorso, il valore è. <code>s3.amazonaws.com</code> Per le richieste in stile virtuale, il valore è. <code>BucketName.s3.amazonaws.com</code> Per ulteriori informazioni, consulta Virtual Hosting nella Amazon Simple Storage Service User Guide. Questa intestazione è obbligatoria per HTTP 1.1 (la maggior parte dei toolkit aggiunge questa intestazione automaticamente); opzionale per le richieste. HTTP/1.0
Origin	Un endpoint che specifica il nome del server del richiedente iniziale.
x-amz-content-sha256	Quando si utilizza la versione 4 della firma per autenticare la richiesta, questa intestazione fornisce un hash del payload della richiesta. Per ulteriori informazioni, consulta Calcolo delle firme per l'intestazione di autorizzazione: trasferimento del payload in un unico blocco (AWS Signature (versione 4)). Quando si carica un oggetto in blocchi, si imposta il valore su <code>STREAMING-AWS4-HMAC-SHA256-PAYLOAD</code> le intestazioni e che non è presente alcun payload. Per ulteriori informazioni, consulta Calcoli della firma per l'intestazione dell'autorizzazione: trasferimento del payload in più blocchi (caricamento in blocchi) (AWS Signature (versione 4)).

Header Name (Nome intestazione)	Description
x-amz-date	Data utilizzata per creare la firma nell'intestazione di autorizzazione. Deve essere nel formato di base ISO 8601 (YYYYMMDD'T'HHMMSS'Z'). Ad esempio, date/time 20170210T120000Z è valido x-amz-date per l'uso con Amazon S3. x-amz-date è facoltativo per tutte le richieste; può essere utilizzato per sovrascrivere la data utilizzata per la firma delle richieste. Se l'intestazione Date è specificata nel formato di base ISO 8601, l'intestazione x-amz-date non è necessaria. Quando x-amz-date è presente, sovrascrive sempre il valore dell'intestazione Date. Per ulteriori informazioni, consulta Gestione delle date in Signature Version 4 nel glossario di Amazon Web Services.

Header Name (Nome intestazione)	Description
x-amz-security-token	Questa intestazione può essere utilizzata nei seguenti scenari: • Per fornire token di sicurezza per DevPay le operazioni di Amazon: ogni richiesta che utilizza Amazon DevPay richiede due x-amz-security-token intestazioni: una per il token del prodotto e una per il token utente. Quando Amazon S3 riceve una richiesta autenticata, confronta la firma calcolata con la firma fornita. Le intestazioni multivalore formattate in modo errato utilizzate per calcolare una firma possono causare problemi di autenticazione. • Per fornire un token di sicurezza quando si utilizzano credenziali di sicurezza temporanee: quando si effettuano richieste utilizzando le credenziali di sicurezza temporanee ottenute da IAM, è necessario fornire un token di sicurezza utilizzando questa intestazione. Per saperne di più sulle credenziali di sicurezza temporanee, consulta Making Requests. https://docs.aws.amazon.com/AmazonS3/latest/userguide/MakingRequests.html Questa intestazione è obbligatoria per le richieste che utilizzano Amazon DevPay e le richieste firmate utilizzando credenziali di sicurezza temporanee.

Intestazioni di risposta comuni

La tabella seguente descrive le intestazioni di risposta comuni alla maggior parte delle risposte di Amazon S3.

Nome	Description
Access-Control-Allow-Credentials	Un valore booleano che determina se il server consente alle richieste CORS di contenere credenziali. Se l'intestazione della Access-Control-Allow-Origin richiesta è impostata su, l'intestazione della Access-Control-Allow-Credentials risposta verrà omessa, altrimenti viene impostata su quando la valutazione CORS ha esito positivo. '*' true Tipo: Booleano Impostazione predefinita: nessuna
Access-Control-Allow-Headers	Un elenco di intestazioni HTTP consentite per le richieste CORS. L'intestazione della Access-Control-Allow-Headers risposta viene restituita in caso di esito positivo delle valutazioni CORS e specifica esplicitamente tutte le opzioni consentite. Access-Control-Request-Headers Tipo: stringa Impostazione predefinita: nessuna
Access-Control-Allow-Methods	Un elenco che specifica quali metodi HTTP sono consentiti. Amazon S3 consentirà le richieste CORS provenienti dai metodi CORS consentiti solo se la valutazione CORS ha esito positivo. Tipo: stringa Impostazione predefinita: nessuna
Access-Control-Allow-Origin	La posizione dell'origine consentita. Amazon S3 invierà l'intestazione della Access-Control-Allow-Origin risposta solo quando la valutazione CORS avrà esito positivo. Se l'origine della richiesta corrisponde alle '*' origini consentite della configurazione CORS, '*' viene restituita in questa intestazione della risposta anziché nell'origine originale.

Nome	Description
	Tipo: stringa Impostazione predefinita: nessuna
Access-Control-Expose-Headers	Un elenco che consente a un server di identificare un'intestazione di risposta che espone l'accesso alle applicazioni quando la valutazione CORS ha esito positivo. Tipo: stringa Impostazione predefinita: nessuna
Access-Control-Max-Age	Il tempo in secondi in cui il browser può memorizzare nella cache la risposta a una richiesta preflight CORS identificata dalla risorsa, dal metodo HTTP e dall'origine. L'intestazione della Access-Control-Max-Age risposta viene restituita solo quando la valutazione CORS ha esito positivo. Tipo: numero intero Impostazione predefinita: nessuna
Vary	Un elenco che indica da quali intestazioni di richiesta varia il risultato della valutazione CORS. L'intestazione della Vary risposta viene restituita solo quando la valutazione CORS ha esito positivo. Tipo: stringa Impostazione predefinita: nessuna
Content-Length	La lunghezza in byte del corpo della risposta. Tipo: stringa Impostazione predefinita: nessuna

Nome	Description
Content-Type	Il tipo MIME del contenuto. Ad esempio, Content-Type: text/html; charset=utf-8 . Tipo: stringa Impostazione predefinita: nessuna
Connection	Un valore che specifica se la connessione al server è aperta o chiusa. Tipo: Enum Valori validi: open close Impostazione predefinita: nessuna
Date	La data e l'ora in cui Amazon S3 ha risposto; ad esempio, mercoledì 1 marzo 2006 12:00:00 GMT. Tipo: stringa Impostazione predefinita: nessuna

Nome	Description
ETag	L'entity tag (ETag) rappresenta una versione specifica dell'oggetto. L'ETag riflette solo i cambiamenti ai contenuti di un oggetto, non i suoi metadata. L'ETag potrebbe o meno essere un digest MD5 dei dati dell'oggetto. Che lo sia o meno dipende da come l'oggetto è stato creato e da come è crittografato, come segue: • Oggetti creati tramite Console di gestione AWS o dall'operazione PUT ObjectPOST Object, or: • Gli oggetti in testo normale o crittografati tramite crittografia lato server con chiavi gestite da Amazon S3 (SSE-S3) hanno ETag che sono un digest MD5 dei loro dati. • Gli oggetti crittografati mediante crittografia lato server con chiavi (SSE-C) o () chiavi () fornite dal cliente hanno ETag che non sono un AWS Key Management Service digest AWS KMS MD5 dei relativi dati oggetto. SSE-KMS • Gli oggetti creati dall'operazione Multipart Upload o Upload Part Copy hanno ETag che non sono digest MD5, indipendentemente dal metodo di crittografia. Tipo: stringa
Server	Il nome del server che ha creato la risposta. Tipo: stringa Impostazione predefinita: AmazonS3
x-amz-delete-marker	Un valore che specifica se l'oggetto restituito era (true) o meno (false) un indicatore di eliminazione. Tipo: Booleano Valori validi: true false Impostazione predefinita: false

Nome	Description
<code>x-amz-id-2</code>	Un token speciale che viene utilizzato insieme all'<code>x-amz-request-id</code> intestazione per aiutare AWS a risolvere i problemi. Per informazioni sull' Supporto utilizzo di questi ID di richiesta, consulta Risoluzione dei problemi di Amazon S3. Tipo: stringa Impostazione predefinita: nessuna
<code>x-amz-request-id</code>	Un valore creato da Amazon S3 che identifica in modo univoco la richiesta . Questo valore viene utilizzato insieme all'<code>x-amz-id-2</code> intestazione per aiutare a risolvere i problemi. AWS Per informazioni sull' Supporto utilizzo di questi ID di richiesta, consulta Troubleshooting Amazon S3. Tipo: stringa Impostazione predefinita: nessuna
<code>x-amz-server-side-encryption</code>	L'algoritmo di crittografia lato server utilizzato per archiviare questo oggetto in Amazon S3 (ad esempio,). AES256 aws:kms Valori validi: AES256 aws:kms
<code>x-amz-version-id</code>	La versione dell'oggetto. Quando abiliti il controllo delle versioni, Amazon S3 genera un numero casuale per gli oggetti aggiunti a un bucket. Il valore è UTF-8 codificato e pronto per l'URL. Quando si crea PUT un oggetto in un bucket in cui il controllo delle versioni è stato sospeso, l'ID della versione è sempre. null Tipo: stringa Valori validi: null qualsiasi stringa URL-ready codificata UTF-8 Di default: null

Risposte agli errori

Questa sezione fornisce informazioni di riferimento sugli errori di Amazon S3.

Note

- In generale, ai proprietari di bucket S3 vengono addebitati i costi associati alle richieste con risposte HTTP 200 OK andate a buon fine e risposte di errore del client HTTP 4XX. Ai proprietari di bucket non vengono addebitati i costi per le risposte di errore del server HTTP 5XX, come ad esempio gli errori HTTP 503 Slow Down. Per ulteriori informazioni sui codici di errore S3 in HTTP 3XX e sui codici di 4XX stato non [fatturati, consulta le risposte agli errori di Billing for Amazon S3](#) nella Amazon S3 User Guide. Per ulteriori informazioni sulle spese di fatturazione se il bucket è configurato come bucket Requester Pays, consulta [How Requester Pays charges work nella Guida per l'utente di Amazon S3](#).
- Il supporto di SOAP su HTTP non viene più utilizzato ma è ancora disponibile su HTTPS. Le nuove funzioni di Amazon S3 non sono supportate per SOAP. Invece di usare SOAP, ti consigliamo di utilizzare l'API REST o gli SDK. AWS

Argomenti

- [Risposte agli errori REST](#)
- [Elenco dei codici di errore](#)
- [Elenco dei codici di errore SELECT Object Content](#)
- [Elenco dei codici di errore Replication-related](#)
- [Elenco dei codici di errore Tagging-related](#)
- [Elenco dei codici di errore relativi alle annotazioni di Amazon S3](#)
- [Elenco dei codici di errore di Amazon S3 on Outposts](#)
- [Elenco dei codici di errore di Amazon S3 Storage Lens](#)
- [Elenco dei codici di errore di Amazon S3 Object Lambda](#)
- [Elenco dei codici di errore asincroni di Amazon S3](#)
- [Elenco dei codici di errore di Amazon S3 Access Grants](#)
- [Elenco dei codici di FSx-related errore Amazon](#)
- [Elenco dei codici di errore di Amazon S3 Tables](#)

- [Best Practice per gli errori Amazon S3](#)

Risposte agli errori REST

Quando si verifica un errore, le informazioni sull'intestazione contengono quanto segue:

- Content-Type: application/xml
- Un codice di stato HTTP 3xx, 4xx o 5xx adeguato

Il corpo della risposta contiene anche informazioni sull'errore. Il seguente esempio di risposta all'errore mostra la struttura degli elementi di risposta comuni a tutte le risposte di errore REST.

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>NoSuchKey</Code>
  <Message>The resource you requested does not exist</Message>
  <Resource>/mybucket/myfoto.jpg</Resource>
  <RequestId>4442587FB7D0A2F9</RequestId>
</Error>
```

La tabella seguente illustra gli elementi di risposta agli errori REST.

Nome	Description
Code	Il codice di errore è una stringa che identifica in modo univoco una condizione di errore. Ha lo scopo di essere letta e compresa dai programmi che rilevano e gestiscono gli errori in base al loro tipo. Per ulteriori informazioni, vedere Elenco dei codici di errore. Tipo: stringa Antenato: Error
Error	Contenitore per tutti gli elementi di errore. Tipo: Container Antenato: Nessuno

Nome	Description
Message	Il messaggio di errore contiene una descrizione generica della condizione di errore in inglese. È destinato a interlocutori umani. I programmi semplici visualizzano il messaggio direttamente all'utente finale se si verifica una condizione di errore che non conoscono o che non sono interessati a gestire. I programmi sofisticati, con una gestione degli errori più completa e una vera e propria internazionalizzazione, più probabilmente ignoreranno il messaggio di errore. Tipo: stringa Antenato: Error
RequestId	ID della richiesta associata all'errore. Tipo: stringa Antenato: Error
Resource	Il bucket o l'oggetto coinvolto nell'errore. Tipo: stringa Antenato: Error

Molte risposte di errore contengono ulteriori dati strutturati destinati a essere letti e compresi da uno sviluppatore che tenti di diagnosticare gli errori di programmazione. Ad esempio, se invii un' Content-MD5 intestazione con una richiesta REST PUT che non corrisponde al digest calcolato sul server, ricevi un errore. BadDigest La risposta all'errore include anche come elementi di dettaglio il digest calcolato dal server e il digest che hai detto al server di aspettarsi. Durante lo sviluppo è possibile utilizzare queste informazioni per la diagnostica dell'errore. In produzione, un programma con un comportamento corretto probabilmente includerebbe queste informazioni in un registro di errore.

Per informazioni sugli elementi di risposta generali, vai a Risposte agli [errori](#).

Elenco dei codici di errore

La tabella seguente elenca i codici di errore di Amazon S3.

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
AccessControlListNotSupported	Il bucket non consente gli ACL.	400 Bad Request	Client
AccessDenied	Accesso negato	403 Forbidden	Client
AccessPointAlreadyOwnedByYou	Nel tuo account esiste già un punto di accesso con un nome identico.	409 Conflict	Client
AccountProblem	C'è un problema nel tuo Account AWS account che impedisce il corretto completamento dell'operazione. Per ulteriore assistenza, vedere Contattaci .	403 Forbidden	Client
AllAccessDisabled	Tutti gli accessi a questa risorsa Amazon S3 sono stati disabilitati. Per ulteriore assistenza, vedere Contattaci .	403 Forbidden	Client
AmbiguousGrantByEmailAddress	L'indirizzo email che hai fornito è associato a più di un account.	400 Bad Request	Client
AuthorizationHeaderMalformed	L'intestazione di autorizzazione fornita non è valida.	400 Bad Request	N/A

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
AuthorizationQueryParametersError	I parametri della query di autorizzazione forniti non sono validi.	400 Bad Request	N/A
BadDigest	Il valore Content-MD5 o il checksum specificato non corrisponde a quello ricevuto dal server.	400 Bad Request	Client
BucketAlreadyExists	Il nome del bucket richiesto non è disponibile. Lo spazio dei nomi del bucket è condiviso da tutti gli utenti del sistema. Specificare un nome diverso e riprovare.	409 Conflict	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
BucketAlreadyOwnedByYou	Il bucket che hai provato a creare esiste già e tu ne sei il proprietario. Amazon S3 restituisce questo errore in tutti i Regioni AWS casi tranne che nella regione Stati Uniti orientali (Virginia settentrionale) (us-east-1). Per motivi di compatibilità con le versioni precedenti, se ricrei un bucket esistente che già possiedi in us-east-1, Amazon S3 restituisce 200 OK e reimposta gli ACL (bucket access control list). Per Amazon S3 on Outposts, il bucket che hai provato a creare esiste già nel tuo Outpost e tu ne sei il proprietario.	409 Conflict tutte le regioni tranne us-east-1)	Client
BucketHasAccessPointsAttached	Al bucket che hai cercato di eliminare sono collegati dei punti di accesso. Elimina i punti di accesso prima di eliminare il bucket.	400 Bad Request	Client
BucketNotEmpty	Il bucket che hai cercato di eliminare non è vuoto.	409 Conflict	Client
ClientTokenConflict	Il token di idempotenza dell' Multi-Region Access Point è già stato utilizzato per una richiesta diversa.	409 Conflict	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
ConnectionClosedByRequester	Restituito al chiamante originale quando si verifica un errore durante la lettura del testo. WriteGetObjectResponse	400 Bad Request	Client
ConditionalRequestConflict	Si è verificata un'operazione in conflitto. In caso di utilizzoPutObject , puoi riprovare la richiesta. Se si utilizza il caricamento in più parti, è necessario avviare un'altra CreateMultipartUpload richiesta e caricare nuovamente ogni parte.	409 Conflict	Client
CredentialsNotSupported	Questa richiesta non supporta le credenziali.	400 Bad Request	Client
CrossLocationLoggingProhibited	Cross-Region la registrazione non è consentita. I bucket contenuti in uno Regione AWS non possono registrare le informazioni in un bucket in un'altra regione.	403 Forbidden	Client
DeviceNotActiveError	Il dispositivo non è attualmente attivo.	400 Bad Request	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
EndpointNotFound	Indirizzare le richieste all'endpoint corretto.	400 Bad Request	Client
EntityTooSmall	Il caricamento proposto è inferiore alla dimensione minima consentita dell'oggetto.	400 Bad Request	Client
EntityTooLarge	Il caricamento proposto supera la dimensione massima consentita dell'oggetto. Per ulteriori informazioni, consulta gli endpoint e le quote di Amazon Simple Storage Service nel . Riferimenti generali di AWS	400 Bad Request	Client
	Il download proposto supera la dimensione massima consentita.	405 Method Not Allowed	Client
ExpiredToken	Il token fornito è scaduto.	400 Bad Request	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
<code>IllegalLocationConstraintException</code>	Questo errore può verificarsi per i seguenti motivi: • Si sta tentando di accedere a un bucket da una Regione diversa da quella in cui esiste il bucket. • Si tenta di creare un bucket con un vincolo di posizione che corrisponde a una regione diversa dall'endpoint regionale a cui è stata inviata la richiesta.	400 Bad Request	Client
<code>IllegalVersioningConfigurationException</code>	La configurazione del controllo delle versioni specificata nella richiesta non è valida.	400 Bad Request	Client
<code>IncompleteBody</code>	Non hai fornito il numero di byte specificato dall'intestazione Content-Length HTTP.	400 Bad Request	Client
<code>IncorrectEndpoint</code>	Il bucket specificato esiste in un'altra Regione. Indirizzare le richieste all'endpoint corretto.	400 Bad Request	Client
<code>IncorrectNumberOfFilesInPostRequest</code>	POST richiede esattamente un caricamento di file per richiesta.	400 Bad Request	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
InlineDataTooLarge	I dati in linea superano la dimensione massima consentita.	400 Bad Request	Client
InternalError	Si è verificato un errore interno. Riprovare.	500 Internal Server Error	Server
InvalidAccessKeyId	L'ID della chiave di AWS accesso che hai fornito non esiste nei nostri registri.	403 Forbidden	Client
InvalidAccessPoint	Il nome o l'account del punto di accesso specificato non è valido.	400 Bad Request	Client
InvalidAccessPointAliasError	Il nome alias del punto di accesso specificato non è valido.	400 Bad Request	Client
InvalidAddressingHeader	È necessario specificare il ruolo Anonymous.	N/A	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
InvalidArgument	Questo errore può verificarsi per i seguenti motivi: - Viene effettuata una ListBuckets richiesta a un endpoint regionale diverso dalla regione specificata nel <code>bucket-region</code> parametro. - L'argomento specificato non è valido. - Nella richiesta manca un'intestazione obbligatoria. - L'argomento specificato è incompleto o nel formato errato. - L'argomento specificato deve avere una lunghezza maggiore o uguale a 3.	400 Bad Request	Client
InvalidBucketAclWithObjectOwnership	Il bucket non può avere ACL impostati con ObjectOwnership l'impostazione's. BucketOwnerEnforced	400 Bad Request	Client
InvalidBucketName	Il bucket specificato non è valido.	400 Bad Request	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
InvalidBucketOwnerAWSAccountID	Il valore del parametro del proprietario del bucket previsto deve essere un Account AWS ID.	400 Bad Request	Client
InvalidBucketState	La richiesta non è valida per lo stato corrente del bucket.	409 Conflict	Client
InvalidDigest	Il valore Content-MD5 o il checksum specificato non è valido.	400 Bad Request	Client
InvalidEncryptionAlgorithmError	La richiesta di crittografia specificata non è valida. Il valore valido è AES256.	400 Bad Request	Client
InvalidHostHeader	Le intestazioni dell'host fornite nella richiesta utilizzano uno stile di indirizzamento errato.	400 Bad Request	Client
InvalidHttpMethod	La richiesta è stata effettuata utilizzando un metodo HTTP imprevisto.	400 Bad Request	Client
InvalidLocationConstraint	Il vincolo di posizione (Regione) specificato non è valido. Per ulteriori informazioni sulla selezione di una regione per i bucket, vedi Panoramica dei <code>bucket</code> .	400 Bad Request	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
InvalidObjectState	L'operazione non è valida per lo stato corrente dell'oggetto.	403 Forbidden	Client
InvalidPart	Non è stato possibile trovare una o più parti specificate. La parte potrebbe non essere stata caricata o il tag di entità specificato potrebbe non corrispondere al tag di entità della parte.	400 Bad Request	Client
InvalidPartOrder	L'elenco delle parti non era in ordine crescente. L'elenco delle parti deve essere specificato in ordine di numero di parte.	400 Bad Request	Client
InvalidPayer	Tutti gli accessi a questo oggetto sono stati disabilitati. Per ulteriore assistenza, vedere Contattaci .	403 Forbidden	Client
InvalidPolicyDocument	Il contenuto del modulo non soddisfa le condizioni specificate nel documento programmatico.	400 Bad Request	Client
InvalidRange	L'intervallo richiesto non può essere soddisfatto.	416 Impossible to attendersi all'inter vallo richiesto	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
InvalidRegion	Hai tentato di creare un punto di Multi-Region accesso in una regione per la quale non hai aderito.	403 Non consentito	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
InvalidRequest	Questo errore può verificarsi per i seguenti motivi: • Una ListBuckets richiesta senza pagina viene effettuata da un account con una quota di bucket per uso generico approvata superiore a 10.000. È necessari o effettuare richieste suddivise per pagina per elencare i bucket in un account con più di 10.000 bucket. • La richiesta utilizza la versione della firma errata. Utilizzare AWS4-HMAC-SHA256 (Signatur e Version 4). • Un punto di accesso può essere creato solo per un bucket esistente. • Il punto di accesso non si trova in uno stato in cui può essere eliminato. • Un punto di accesso può essere elencato solo per un bucket esistente.	400 Bad Request	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
	• Il token successivo non è valido. • È necessario specificare almeno un'operazione in una regola del ciclo di vita. • È necessario specificare almeno una regola del ciclo di vita. • Il numero di regole del ciclo di vita non deve superare il limite consentito di 1.000 regole. • L'intervallo per il parametro <code>MaxResults</code> non è valido. • Le richieste SOAP devono essere effettuate tramite una connessione HTTPS. • Amazon S3 Transfer Acceleration non è supportato per i bucket con nomi non conformi a DNS. • Amazon S3 Transfer Acceleration non è supportato per i bucket con punti (.) nei nomi. •		

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
	L'endpoint Amazon S3 Transfer Acceleration supporta solo richieste in stile virtuale. • Amazon S3 Transfer Acceleration non è configurato in questo bucket. • Amazon S3 Transfer Acceleration è disabilitato in questo bucket. • Amazon S3 Transfer Acceleration non è supportato in questo bucket. Per assistenza, contattar e il Supporto. • Amazon S3 Transfer Acceleration non può essere abilitato in questo bucket. Per assistenza, contattar e il Supporto. • Valori in conflitto forniti nelle intestazioni HTTP e nei parametri di query. • Valori in conflitto forniti nelle intestazioni HTTP e nei campi del modulo POST. •		

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
	CopyObject richiesta effettuata su oggetti di dimensioni superiori a 5 GB.		
InvalidSessionException	Restituito se la sessione non esiste più perché è andata in timeout o scaduta.	400 Bad Request	Client
InvalidSignature	La firma della richiesta calcolata dal server non corrisponde alla firma fornita dall'utente. Controlla la tua chiave di accesso AWS segreta e il metodo di firma. Per ulteriori informazioni, consulta Signing and authenticating REST requests .	400 Bad Request	Client
InvalidSecurity	Le credenziali di sicurezza fornite non sono valide.	403 Forbidden	Client
InvalidSOAPRequest	Il corpo della richiesta SOAP non è valido.	400 Bad Request	Client
InvalidStorageClass	La classe di storage specificata non è valida.	400 Bad Request	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
InvalidTargetBucketForLogging	Il bucket di destinazione per la registrazione non esiste, non è di tua proprietà o non dispone delle autorizzazioni appropriate per il gruppo di consegna dei log.	400 Bad Request	Client
InvalidToken	Il token fornito ha un formato errato o comunque non è valido.	400 Bad Request	Client
InvalidURI	L'URI specificato non può essere analizzato.	400 Bad Request	Client
KeyTooLongError	La chiave è troppo lunga.	400 Bad Request	Client
KMS.DisabledException	La richiesta è stata rifiutata perché la chiave KMS specificata non è abilitata.	400 Bad Request	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
KMS.InvalidKeyUsageException	La richiesta è stata rifiutata per uno dei seguenti motivi: • Il KeyUsage valore della chiave KMS è incompatibile con l'operazione API. • L'algoritmo di crittografia o l'algoritmo di firma specificato per l'operazione non è compatibile con il tipo di materiale chiave contenuto nella chiave KMS (). KeySpec Per crittografare, decrittografare, ricrittografare e generare chiavi di dati, deve essere ENCRYPT_DECRYPT. KeyUsage Per firmare e verificare i messaggi, deve essere SIGN_VERIFY. KeyUsage Per generare e verificare i codici di autenticazione dei messaggi (MAC), deve essere GENERATE_VERIFY_MAC. KeyUsage Per derivare i segreti degli accordi chiave, deve essere KEY_AGREEMENT. KeyUsage Per trovare il codice KeyUsage di una chiave KMS, usa l'operazione. DescribeKey	400 Bad Request	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
	Per trovare gli algoritmi di crittografia o firma supportati per una particolare chiave KMS, usa l'operazione. DescribeKey		

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
KMS.KMSInvalidStateException	La richiesta è stata rifiutata perché lo stato della risorsa specificata non è valido per questa richiesta. Questa eccezione indica una delle seguenti situazioni: - Lo stato della chiave KMS non è compatibile con l'operazione. Per trovare lo stato della chiave, usa l' <code>DescribeKey</code> operazione. Per saperne di più su quali sono gli stati delle chiavi compatibili con ciascuna operazione KMS, consulta Key states of AWS KMS keys nella Guida per gli sviluppatori del AWS Key Management Service . - Per le operazioni di crittografia sulle chiavi KMS negli archivi di chiavi personalizzati, questa eccezione rappresenta un errore generale con molte possibili cause. Per identificare la causa, vedere il messaggio di errore che accompagna l'eccezione.	400 Bad Request	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
<code>KMS.NotFoundException</code>	La richiesta è stata rifiutata perché non è possibile trovare l'entità o la risorsa specificata.	400 Bad Request	Client
<code>MalformedACLError</code>	L'ACL fornito non è in un formato valido o non è stato convalidato in base al nostro schema pubblicato.	400 Bad Request	Client
<code>MalformedPOSTRequest</code>	Il corpo della tua richiesta POST non è ben formato multipart/form - data.	400 Bad Request	Client
<code>MalformedXML</code>	Il documento XML fornito non è in un formato valido o non è stato convalidato in base al nostro schema pubblicato.	400 Bad Request	Client
<code>MaxMessageLengthExceeded</code>	La tua richiesta era troppo grande.	400 Bad Request	Client
<code>MaxPostPreDataLengthExceededError</code>	I campi della richiesta POST che precedono il file di caricamento sono troppo grandi.	400 Bad Request	Client
<code>MetadataTooLarge</code>	Le intestazioni dei metadati superano la dimensione massima consentita per i metadati.	400 Bad Request	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
MethodNotAllowed	Il metodo specificato non è consentito su questa risorsa.	405 Method Not Allowed	Client
MissingAttachment	Era previsto un allegato SOAP, ma non è stato trovato.	400 Richiesta non valida	Client
MissingAuthenticationToken	La richiesta non è firmata.	403 Non consentito	Client
MissingContentLength	È necessario fornire l'intestazione Content-Length HTTP.	411 Length Required	Client
MissingRequestBodyError	È stato inviato un documento XML vuoto come richiesta.	400 Bad Request	Client
MissingSecurityElement	Nella richiesta SOAP 1.1 manca un elemento di sicurezza.	400 Bad Request	Client
MissingSecurityHeader	Nella richiesta manca un'intestazione obbligatoria.	400 Bad Request	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
NoLoggingStatusForKey	Non esiste una risorsa secondaria dello stato di registrazione per una chiave.	400 Bad Request	Client
NoSuchAsyncRequest	La richiesta specificata non è stata trovata.	404 Not Found	Client
NoSuchBucket	Il bucket specificato non esiste.	404 Not Found	Client
NoSuchBucketPolicy	Il bucket specificato non ha una policy per il bucket.	404 Not Found	Client
NoSuchCORSConfiguration	Il bucket specificato non ha una configurazione CORS.	404 Not Found	Client
NoSuchKey	La chiave specificata non esiste.	404 Not Found	Client
NoSuchLifecycleConfiguration	La configurazione del ciclo di vita specificata non esiste.	404 Not Found	Client
NoSuchMultiRegionAccessPoint	L' Multi-Region Access Point specificato non esiste.	404 Not Found	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
NoSuchObjectLockConfiguration	L'oggetto specificato non ha una ObjectLock configurazione.	404 Not Found	Client
NoSuchWebsiteConfiguration	Il bucket specificato non ha una configurazione del sito web.	404 Not Found	Client
NoSuchTagSet	Il tag specificato non esiste.	404 Not Found	Client
NoSuchUpload	Il caricamento in più parti specificato non esiste. L'ID di caricamento potrebbe non essere valido oppure il caricamento in più parti potrebbe essere stato interrotto o completato.	404 Not Found	Client
NoSuchVersion	L'ID della versione specificato nella richiesta non corrisponde a una versione esistente.	404 Not Found	Client
NotDeviceOwnerError	Il dispositivo che ha generato il token non è di proprietà dell'utente autenticato.	400 Bad Request	Client
NotImplemented	L'intestazione fornita implica una funzionalità non implementata.	501 Not Implemented	Server

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
NotModified	La risorsa non è stata modificata.	304 Not Modified	Server
NoTransformationDefined	Nessuna trasformazione trovata per questo punto di accesso per le espressioni Lambda dell'oggetto.	404 Not Found	Client
NotSignedUp	L'account non è registrato per il servizio Amazon S3. È necessari o registrarsi prima di poter utilizzar e Amazon S3. Puoi registrarti al seguente URL: https://aws.amazon.com/s3	403 Forbidden	Client
ObjectLockConfigurationNotFoundError	La configurazione Object Lock non esiste per questo bucket.	404 Not Found	Client
OwnershipControlsNotFoundError	I controlli di proprietà del bucket non sono stati trovati.	404 Not Found	Client
OperationAborted	Su questa risorsa è attualmente in corso un'operazione condizionale in conflitto. Riprovare.	409 Conflict	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
PermanentRedirect	Il bucket a cui si sta tentando di accedere deve essere indirizzato tramite l'endpoint specificato. Inviare tutte le richieste future a questo endpoint.	301 Moved Permanently	Client
PermanentRedirectControlError	L'operazione API a cui si sta tentando di accedere deve essere indirizzata tramite l'endpoint specificato. Inviare tutte le richieste future a questo endpoint.	301 Moved Permanently	Client
PreconditionFailed	Almeno una delle condizioni preliminari specificate non è soddisfatta.	412 Precondition Failed	Client
Redirect	Reindirizzamento temporaneo. L'utente viene reindirizzato al bucket mentre il server del sistema dei nomi di dominio (DNS) viene aggiornato.	307 Temporary Redirect	Client
RequestHeaderSectionTooLarge	L'intestazione della richiesta e i parametri di interrogazione utilizzati per far sì che la richiesta superi la dimensione massima consentita.	400 Bad Request	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
RequestIsNotMultipartContent	Una richiesta POST del bucket deve essere di tipo multipart/form enclosure-data.	412 Precondition Failed	Client
RequestTimeout	La connessione socket al server non è stata letta o scritta entro il periodo di timeout.	400 Bad Request	Client
RequestTimeTooSkewed	La differenza tra l'ora della richiesta e l'ora del server è troppo grande.	403 Forbidden	Client
RequestTorrentOfBucketError	Non è consentito richiedere il file torrent di un bucket.	400 Bad Request	Client
ResponseInterrupted	Restituito al chiamante originale quando si verifica un errore durante la lettura del corpo. WriteGetObjectResponse	400 Bad Request	Client
RestoreAlreadyInProgress	Il ripristino dell'oggetto è già in corso.	409 Conflict	Client
ServerSideEncryptionConfigurationNotFound	La configurazione di crittografia sul lato server non è stata trovata.	400 Bad Request	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
ServiceUnavailable	Il servizio non è in grado di gestire la richiesta.	503 Service Unavailable	Server
SignatureDoesNotMatch	La firma della richiesta calcolata dal server non corrisponde alla firma fornita dall'utente. Controlla la tua chiave di accesso AWS segreta e il metodo di firma. Per ulteriori informazioni, vedere Autenticazione REST e Autenticazione SOAP .	403 Forbidden	Client
SlowDown	Riduci il tasso di richieste.	503 Slow Down	Server
503 SlowDown	Rallenta	503 Slow Down	Server
TagPolicyException	La politica sui tag non consente il valore specificato per la seguente chiave di tag.	400 Bad Request	Client
TemporaryRedirect	L'utente viene reindirizzato al bucket mentre il server del sistema dei nomi di dominio (DNS) viene aggiornato.	307 Temporary Redirect	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
TokenCodeInvalidError	Il codice and/or token con numero di serie che hai fornito non è valido.	400 Bad Request	Client
TokenRefreshRequired	Il token fornito deve essere aggiornato.	400 Bad Request	Client
TooManyAccessPoints	Hai tentato di creare più punti di accesso di quelli consentiti per un account. Per ulteriori informazioni, consulta gli endpoint e le quote di Amazon Simple Storage Service nel . Riferimenti generali di AWS	400 Bad Request	Client
TooManyBuckets	Hai tentato di creare più bucket di quelli consentiti per un account. Per ulteriori informazioni, consulta gli endpoint e le quote di Amazon Simple Storage Service nel. Riferimenti generali di AWS	400 Bad Request	Client
TooManyMultiRegionAccessPointregionsError	Hai tentato di creare un punto di Multi-Region accesso con più regioni di quelle consentite per un account. Per ulteriori informazioni, consulta gli endpoint e le quote di Amazon Simple Storage Service nel . Riferimenti generali di AWS	400 Bad Request	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
TooManyMultiRegionAccessPoints	Hai tentato di creare più punti di Multi-Region accesso di quelli consentiti per un account. Per ulteriori informazioni, consulta gli endpoint e le quote di Amazon Simple Storage Service nel Riferimenti generali di AWS	400 Bad Request	Client
UnauthorizedAccessError	Applicabile solo nelle Regioni della Cina. Restituito quando viene effettuata una richiesta a un bucket che non dispone di una licenza ICP. Per ulteriori informazioni, consulta ICP Recordal .	403 Forbidden	Client
UnexpectedContent	Questa richiesta contiene contenuti non supportati.	400 Bad Request	Client
UnexpectedIPError	Applicabile solo nelle Regioni della Cina. La richiesta è stata rifiutata perché l'IP era imprevisto.	403 Forbidden	Client
UnsupportedArgument	La richiesta contiene un argomento non supportato.	400 Bad Request	Client
UnsupportedSignature	La richiesta fornita è firmata con una versione del token STS non supportata o la versione della firma non è supportata.	400 Bad Request	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
UnresolvableGrantByEmailAddress	L'indirizzo email che hai fornito non corrisponde a nessun account registrato.	400 Bad Request	Client
UserKeyMustBeSpecified	La richiesta POST del bucket deve contenere il nome di campo specificato. Se è specificato, controllare l'ordine dei campi.	400 Bad Request	Client
NoSuchAccessPoint	Il punto di accesso specificato non esiste.	404 Not Found	Client
InvalidTag	La richiesta contiene un tag inserito non valido. Ad esempio, la richiesta potrebbe contenere chiavi duplicate , chiavi o valori troppo lunghi oppure tag di sistema.	400 Bad Request	Client
MalformedPolicy	La tua polizza contiene un principio non valido.	400 Bad Request	Client

Elenco dei codici di errore SELECT Object Content

Important

Amazon S3 Select non è più disponibile per i nuovi clienti. I clienti esistenti di Amazon S3 Select possono continuare a utilizzare la funzionalità come di consueto. [Ulteriori informazioni](#)

La tabella seguente contiene errori speciali che SELECT Object Content potrebbero verificarsi. Per informazioni generali sugli errori di Amazon S3 e un elenco di codici di errore, consulta [Risposte agli errori](#).

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
AmbiguousFieldName	Il nome del campo corrisponde a più campi del file. Controlla l'espressione SQL e il file e riprova.	400	Client
Busy	Il servizio non è disponibile. Riprova più tardi.	503	Client
CastFailed	Un tentativo di conversione da un tipo di dati a un altro utilizzando l'espressione SQL CAST non è riuscito.	400	Client
ColumnTooLong	La lunghezza di una colonna nel risultato è superiore a maxCharsPerColumn 1 MB.	400	Client
CSVEscapingRecordDelimiter	Nel file è stato trovato un delimitatore di record tra virgolette. Per consentire i delimitatori di record tra virgolette, imposta su. AllowQuotedRecordDelimiter 'TRUE'	400	Client
CSVParsingError	Si è verificato un errore durante l'analisi del file CSV. Controlla il file e riprova.	400	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
CSVUnescapedQuote	È stata trovata una citazione senza escape durante l'analisi del file CSV. Per consentire i delimitatori di record tra virgolette, imposta su. AllowQuotedRecordDelimiter 'TRUE'	400	Client
EmptyRequestBody	Il corpo della richiesta non può essere vuoto.	400	Client
EvaluatorBindingDoesNotExist	Un nome di colonna o un percorso fornito non esiste nell'espressione SQL.	400	Client
EvaluatorInvalidArguments	Il numero di argomenti nella chiamata alla funzione nell'espressione SQL è errato.	400	Client
EvaluatorInvalidTimestampFormatPattern	La stringa di formato timestamp nell'espressione SQL non è valida.	400	Client
EvaluatorInvalidTimestampFormatPatternSymbol	Il modello di formato timestamp contiene un simbolo nell'espressione SQL che non è valido.	400	Client
EvaluatorInvalidTimestampFormatPatternSymbolForParsing	Il modello di formato timestamp contiene un simbolo di formato valido che non può essere applicato all'analisi del timestamp nell'espressione SQL.	400	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
<code>EvaluatorInvalidTimestampFormatPatternToken</code>	Il modello di formato timestamp contiene un token nell'espressione SQL che non è valido.	400	Client
<code>EvaluatorLikePatternInvalidEscapeSequence</code>	Un argomento fornito all'LIKE espressione non era valido.	400	Client
<code>EvaluatorNegativeLimit</code>	LIMIT non deve essere negativo.	400	Client
<code>EvaluatorTimestampFormatPatternDuplicateFields</code>	Il modello di formato timestamp contiene più specificatori di formato che rappresentano il campo timestamp nell'espressione SQL.	400	Client
<code>EvaluatorTimestampFormatPatternHourClockAmPmMismatch</code>	Il modello di formato timestamp contiene un simbolo di formato dell'ora del giorno di 12 ore ma non contiene anche un AM/PM campo, oppure contiene un identificatore di formato dell'ora del giorno di 24 ore e contiene un campo nell'espressione SQL. AM/PM	400	Client
<code>EvaluatorUnterminatedTimestampFormatPatternToken</code>	Il modello di formato timestamp contiene un token non terminato nell'espressione SQL.	400	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
ExpressionTooLong	L'espressione SQL è troppo lunga. La lunghezza massima in byte per un'espressione SQL è 256 KB.	400	Client
ExternalEvalException	L'interrogazione non può essere valutata. Controlla il file e riprova.	400	Client
IllegalSqlFunctionArgument	Nella funzione SQL è stato utilizzato un argomento illegale.	400	Client
IncorrectSqlFunctionArgumentType	Un tipo di argomento errato è stato specificato in una chiamata di funzione nell'espressione SQL.	400	Client
IntegerOverflow	Nell'espressione SQL si è verificato un overflow o underflow di numeri interi.	400	Client
InternalError	Si è verificato un errore interno.	500	Client
InvalidCast	Un tentativo di conversione da un tipo di dati a un altro utilizzando l'espressione SQL CAST non è riuscito.	400	Client
InvalidColumnIndex	L'indice delle colonne nell'espressione SQL non è valido.	400	Client
InvalidCompressionFormat	Il file non è in un formato di compressione supportato. Sono supportati solo GZIP e BZIP2.	400	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
InvalidDataSource	Il tipo di origine dati non è valido. Sono supportati solo CSV, JSON e Parquet.	400	Client
InvalidDataType	L'espressione SQL contiene un tipo di dati non valido.	400	Client
InvalidExpressionType	Il ExpressionType valore non è valido. Sono supportate solo le espressioni SQL.	400	Client
InvalidFileHeaderInfo	Il FileHeaderInfo valore non è valido. Solo NONEUSE, e IGNORE sono supportati.	400	Client
InvalidJsonType	Il JsonType valore non è valido. Solo DOCUMENT e LINES sono supportati.	400	Client
InvalidKeyPath	Il percorso chiave nell'espressione SQL non è valido.	400	Client
InvalidQuoteFields	Il QuoteFields valore non è valido. Solo ALWAYS e ASNEEDED sono supportati.	400	Client
InvalidRequestParameter	Il valore di un parametro nell>SelectRequest elemento non è valido. Controlla la documentazione dell'API del servizio e riprova.	400	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
InvalidScanRange	L'intervallo di scansione fornito non è valido.	400	Client
InvalidTableAlias	L'espressione SQL contiene un alias di tabella non valido.	400	Client
InvalidTextEncoding	Il tipo di codifica non è valido. È supportata solo UTF-8 la codifica.	400	Client
JSONParsingError	Si è verificato un errore durante l'analisi del file JSON. Controlla il file e riprova.	400	Client
LexerInvalidChar	L'espressione SQL contiene un carattere non valido.	400	Client
LexerInvalidIONLiteral	L'espressione SQL contiene un operatore non valido.	400	Client
LexerInvalidLiteral	L'espressione SQL contiene un operatore non valido.	400	Client
LexerInvalidOperator	L'espressione SQL contiene un valore letterale non valido.	400	Client
LikeInvalidInputs	L'argomento fornito alla LIKE clausola nell'espressione SQL non è valido.	400	Client
MalformedXML	L'XML fornito non era ben formato o non è stato convalidato rispetto al nostro schema pubblicato. Controlla la documentazione del servizio e riprova.	400	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
MaxOperatorsExceeded	Impossibile analizzare l'espressione SQL, prova a ridurre la complessità. Ad esempio, riduci il numero di operatori utilizzati.	400	Client
MethodNotAllowed	Il metodo specificato non è consentito su questa risorsa.	405 Method Not Allowed	Client
MissingRequiredParameter	All'SelectRequest entità manca un parametro obbligatorio. Controlla la documentazione del servizio e riprova.	400	Client
MultipleDataSourceUnsupported	Le origini dati multiple non sono supportate.	400	Client
NumberFormatError	Si è verificato un errore durante l'analisi di un numero. Questo errore può essere causato da un sottoflusso o da un overflow di numeri interi.	400	Client
ObjectSerializationConflict	InputSerialization specifica più di un formato (CSV, JSON o Parquet) o OutputSerialization specifica più di un formato (CSV o JSON). Per InputSerialization e OutputSerialization, puoi specificare un solo formato per ciascuno.	400	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
OverMaxColumn	Il numero di colonne nel risultato è maggiore del numero massimo consentito di colonne.	400	Client
OverMaxParquetBlockSize	Il file Parquet è superiore alla dimensione massima del gruppo di righe.	400	Client
OverMaxRecordSize	La lunghezza di un record nell'input o nel risultato è superiore al maxCharsPerRecord limite di 1 MB.	400	Client
ParquetParsingError	Si è verificato un errore durante l'analisi del file Parquet. Controlla il file e riprova.	400	Client
ParquetUnsupportedCompressionCodec	Il codec di compressione Parquet specificato non è supportato.	400	Client
ParseAsteriskIsNotAloneInSelectList	Le altre espressioni non sono consentite nell'SELECT elenco quando * viene utilizzato senza la notazione a punti nell'espressione SQL.	400	Client
ParseCannotMixSqbAndWildcardInSelectList	Non è possibile combinare [] e * nella stessa espressione in un SELECT elenco nell'espressione SQL.	400	Client
ParseCastArity	L'identità dell'espressione SQL non CAST è corretta.	400	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
ParseEmptySelect	L'espressione SQL contiene una SELECT clausola vuota.	400	Client
ParseExpected2TokenTypes	Il token previsto nell'espressione SQL non è stato trovato.	400	Client
ParseExpectedArgumentDelimiter	Il delimitatore di argomento previsto nell'espressione SQL non è stato trovato.	400	Client
ParseExpectedDatePart	La parte relativa alla data prevista nell'espressione SQL non è stata trovata.	400	Client
ParseExpectedExpression	L'espressione SQL prevista non è stata trovata.	400	Client
ParseExpectedIdentifierForAlias	L'identificatore previsto per l'alias nell'espressione SQL non è stato trovato.	400	Client
ParseExpectedIdentifierForAt	L'identificatore previsto per AT il nome nell'espressione SQL non è stato trovato.	400	Client
ParseExpectedIdentifierForGroupName	GROUP non è supportato nell'espressione SQL.	400	Client
ParseExpectedKeyword	La parola chiave prevista nell'espressione SQL non è stata trovata.	400	Client
ParseExpectedLeftParenAfterCast	La parentesi sinistra prevista dopo CAST nell'espressione SQL non è stata trovata.	400	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
ParseExpectedLeftParenBuiltinFunctionCall	La parentesi sinistra prevista nell'espressione SQL non è stata trovata.	400	Client
ParseExpectedLeftParenValueConstructor	La parentesi sinistra prevista nell'espressione SQL non è stata trovata.	400	Client
ParseExpectedMember	L'espressione SQL contiene un uso non supportato di. MEMBER	400	Client
ParseExpectedNumber	Il numero previsto nell'espressione SQL non è stato trovato.	400	Client
ParseExpectedRightParenBuiltinFunctionCall	Il carattere di parentesi destra previsto nell'espressione SQL non è stato trovato.	400	Client
ParseExpectedTokenType	Il token previsto nell'espressione SQL non è stato trovato.	400	Client
ParseExpectedTypeName	Il nome del tipo previsto nell'espressione SQL non è stato trovato.	400	Client
ParseExpectedWhenClause	La WHEN clausola prevista nell'espressione SQL non è stata trovata. CASE non è supportata.	400	Client
ParseInvalidContextForWildcardInSelectList	L'uso di * nell'SELECT elenco nell'espressione SQL non è valido.	400	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
ParseInvalidPathComponent	L'espressione SQL contiene un componente di percorso non valido.	400	Client
ParseInvalidTypeParam	L'espressione SQL contiene un valore di parametro non valido.	400	Client
ParseMalformedJoin	JOIN non è supportata nell'espressione SQL.	400	Client
ParseMissingIdentifierAt	L'identificatore previsto dopo il simbolo @ nell'espressione SQL non è stato trovato.	400	Client
ParseNonUnaryAggregateFunctionCall	Nell'espressione SQL è supportato un solo argomento per le funzioni aggregate.	400	Client
ParseSelectMissingFrom	L'espressione SQL contiene un elemento mancante FROM dopo l'SELECT.	400	Client
ParseUnexpectedKeyword	L'espressione SQL contiene una parola chiave inaspettata.	400	Client
ParseUnexpectedOperator	L'espressione SQL contiene un operatore inaspettato.	400	Client
ParseUnexpectedTerm	L'espressione SQL contiene un termine imprevisto.	400	Client
ParseUnexpectedToken	L'espressione SQL contiene un token inaspettato.	400	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
ParseUnknownOperator	L'espressione SQL contiene un operatore non valido.	400	Client
ParseUnsupportedAlias	L'espressione SQL contiene un uso non supportato di ALIAS.	400	Client
ParseUnsupportedCallWithStar	Nell'espressione SQL è supportato solo il parametro COUNT with (*) as.	400	Client
ParseUnsupportedCase	L'espressione SQL contiene un uso non supportato di CASE.	400	Client
ParseUnsupportedCaseClause	L'espressione SQL contiene un uso non supportato di. CASE	400	Client
ParseUnsupportedLiteralsGroupBy	L'espressione SQL contiene un uso non supportato di. GROUP BY	400	Client
ParseUnsupportedSelect	L'espressione SQL contiene un uso non supportato di. SELECT	400	Client
ParseUnsupportedSyntax	L'espressione SQL contiene una sintassi non supportata.	400	Client
ParseUnsupportedToken	L'espressione SQL contiene un token non supportato.	400	Client
TruncatedInput	Decompressione dell'oggetto non riuscita. Verifica che l'oggetto sia compresso correttamente utilizzando il formato specificato nella richiesta.	400	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
UnauthorizedAccess	Non hai le autorizzazioni necessari e per eseguire questa operazione.	401	Client
UnrecognizedFormatException	Il tipo di record non è valido.	400	Client
UnsupportedFunction	S3 ha rilevato una funzione SQL non supportata.	400	Client
UnsupportedParquetType	Il tipo di parquet specificato non è supportato.	400	Client
UnsupportedRangeHeader	Un'intestazione di intervallo non è supportata per questa operazione.	400	Client
UnsupportedScanRangeInput	Le interrogazioni sull'intervallo di scansione non sono supportate su questo tipo di oggetto.	400	Client
UnsupportedSqlOperation	S3 ha rilevato un'operazione SQL non supportata.	400	Client
UnsupportedSqlStructure	S3 ha rilevato una struttura SQL non supportata. Controlla il riferimento SQL.	400	Client
UnsupportedStorageClass	La classe di archiviazione non è supportata. Solo STANDARD_STANDARD_IA , e le classi ONEZONE_IA di archiviazione sono supportate.	400	Client
UnsupportedSyntax	La sintassi non è valida.	400	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
UnsupportedTypeFor Querying	La query contiene un tipo non supportato per il confronto (ad esempio, verifica che un tipo di colonna Parquet INT96 sia maggiore di 0).	400	Client
ValueParseFailure	Si è verificato un errore di analisi del timestamp nell'espressione SQL.	400	Client

Elenco dei codici di errore Replication-related

La tabella seguente contiene errori speciali che l'Replicationoperazione potrebbe restituire. Per informazioni generali sugli errori di Amazon S3 e un elenco di codici di errore, consulta [Risposte agli errori](#).

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
InvalidArgument	Questo errore può verificarsi per i seguenti motivi: • L'<Account> elemento è vuoto. Deve contenere un ID account valido. • Il Account AWS valore specifica to nell'<Account> elemento	400	Client

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
	deve corrispondere al proprietario del bucket di destinazione. • <code>ReplicationTime-Status</code> deve contenere un valore. • <code>ReplicationTime-ReplicationTimeValue</code> deve contenere un valore. • <code>Replication-ReplicationTimeValue-Minutes</code> il valore deve essere 15. • <code>ReplicationMetrics</code> deve contenere un <code>Status</code>. • <code>ReplicationMetrics</code> deve contenere un <code>EventThreshold</code>. • <code>EventThreshold-ReplicationTimeValue-Minutes</code> il valore deve essere 15. • <code>Rule ID</code> non deve contenere caratteri non ASCII.		

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
InvalidRequest	Questo errore può verificarsi per i seguenti motivi: • L'<Owner>in <AccessControlTranslation> ha un valore, quindi l'<Account> elemento deve essere specificato. • L'<Account> elemento è vuoto. Deve contenere un ID account valido. • La replica destination deve contenere entrambi ReplicationTime o nessuno dei due. Metrics • ReplicationTime e ReplicationMetrics deve avere lo stesso stato. • S3 Replication Time Control (S3 RTC) non è supportato in questo caso. Regione AWS	400	Client
ReplicationConfigurationNotFoundError	Non esiste una configurazione di replica per questo bucket.	404 Not Found (404 Non trovato)	Client

Elenco dei codici di errore Tagging-related

La tabella seguente contiene errori speciali che le `ListTagsForResource` operazioni `TagResource` `UntagResource`, e potrebbero restituire per i gruppi Storage Lens. Per informazioni generali sugli errori generali di Amazon S3 e un elenco di codici di errore, consulta [Risposte agli errori](#).

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
<code>InvalidRequest</code>	L'ARN Regione AWS nella risorsa non corrisponde alla regione specificata in questa richiesta. L' AWS account nell'ARN della risorsa non corrisponde all'ID dell'account specificato in questa richiesta. La AWS partizione in <code>ResourceARN</code> non è valida.	400 Bad Request	Non supportata
<code>InvalidTag</code>	Questa richiesta contiene una chiave o un valore di tag non valido. I caratteri validi includono quanto segue: <code>[a-zA-Z+-=._:/]</code> . Le chiavi dei tag possono contenere fino a 128 caratteri. I valori dei tag possono contenere fino a 256 caratteri. Ci sono chiavi tag duplicate nella tua richiesta. User-defined le chiavi dei tag non possono iniziare con <code>aws:</code> .	400 Bad Request	Non supportata
<code>NoSuchResource</code>	La risorsa specificata non esiste.	404 Not Found	Non supportata

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
TagPolicyException	La politica sui tag non consente il valore specificato per la seguente chiave di tag.	400 Bad Request	Non supportata
TooManyTags	Il numero di tag supera il limite di 50 tag.	400 Bad Request	Non supportata

Elenco dei codici di errore relativi alle annotazioni di Amazon S3

La tabella seguente contiene errori speciali che le operazioni di annotazione (PutObjectAnnotation,, GetObjectAnnotation

ListObjectAnnotationsDeleteObjectAnnotation,UpdateBucketMetadataAnnotationTableCo

e il comportamento correlato alle annotazioni potrebbero restituire. CopyObject Per informazioni generali sugli errori di Amazon S3 e un elenco di codici di errore, consulta. [Risposte agli errori](#)

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
AccessDenied	Accesso negato. Il chiamante non dispone dell'autorizzazione per eseguire l'operazione di annotazione richiesta oppure l'oggetto è protetto da Object Lock in modalità COMPLIANCE.	403 Forbidden	Non supportata
AnnotationLimitExceeded	Il numero di annotazioni dell'oggetto supera il limite (1000).	400 Bad Request	Non supportata
AnnotationNameTooLong	Il nome dell'annotazione è troppo lungo. I nomi delle annotazioni	400 Bad Request	Non supportata

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
	possono avere una lunghezza massima di 512 byte.		
AnnotationRequestConflict	È attualmente in corso un'operazione in conflitto su questa annotazione. Ciò può verificarsi quando un'annotazione di origine viene eliminata durante una copia interregionale.	409 Conflict	Non supportata
EntityTooSmall	Il caricamento proposto è inferiore alla dimensione minima consentita. Il payload delle annotazioni deve essere di almeno 1 byte.	400 Bad Request	Non supportata
InvalidAnnotationName	Il nome dell'annotazione contiene caratteri non validi oppure inizia con un prefisso riservato (o senza distinzione tra maiuscole e minuscole). aws s3 I nomi delle annotazioni possono contenere solo lettere Unicode, numeri, trattino basso (-), punto (.) e trattino underscore (_). . -	400 Bad Request	Non supportata
InvalidArgument	La direttiva sulle annotazioni non è valida. I valori validi sono COPY e EXCLUDE. ForListObjectsAnnotations, max-annotation-results deve essere compreso tra 1 e 1000.	400 Bad Request	Non supportata

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
InvalidPrefix	Il prefisso di annotazione fornito non è valido.	400 Bad Request	Non supportata
InvalidRequest	Questo errore può verificarsi per i seguenti motivi: L'oggetto è crittografato con SSE-C, che non supporta le annotazioni. L'oggetto di origine contiene annotazioni che non possono essere copiate nella destinazione perché la destinazione si trova in una regione non supportata. La richiesta di configurazione della tabella di annotazione non è valida (ad esempio, disabilitazione della tabella mentre si specifica la crittografia, attivazione di una tabella già abilitata, aggiornamento di una tabella in stato CREATING o invio di una richiesta vuota).	400 Bad Request	Non supportata
MethodNotAllowed	Il metodo specificato non è consentito su questa risorsa. Ad esempio, questo errore si verifica quando la richiesta è indirizzata a un indicatore di eliminazione.	405 Method Not Allowed	Non supportata
NoSuchAnnotation	L'annotazione specificata non esiste.	404 Not Found	Non supportata

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
NoSuchKey	La chiave specificata non esiste.	404 Not Found	Non supportata
NoSuchVersion	L'ID della versione specificato nella richiesta non corrisponde a una versione esistente.	404 Not Found	Non supportata
NotImplemented	Le annotazioni non sono supportate per i bucket di directory Amazon S3.	501 Not Implemented	Non supportata
OperationAborted	Su questa risorsa è attualmente in corso un'operazione condizionale in conflitto. Ciò può verificarsi durante modifiche simultanee e o quando il conteggio delle annotazioni di origine o l'ETag cambia durante una copia interregionale. Riprovare.	409 Conflict	Non supportata
PreconditionFailed	Almeno una delle precondizioni specificate non è stata soddisfatta. L' <code>x-amz-object-if-match</code> ETag non corrisponde all'oggetto ETag corrente.	412 Precondition Failed	Non supportata
UnsupportedMediaType	Il payload della richiesta contiene un tipo di file multimediale non supportato. Il payload delle annotazioni deve essere un testo codificato valido. UTF-8	415 Unsupported Media Type	Non supportata

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
UserAnnotationName MustBeSpecified	È necessario specificare un nome utente per l'annotazione.	400 Bad Request	Non supportata

Elenco dei codici di errore di Amazon S3 on Outposts

La tabella seguente contiene errori speciali che un'operazione di Amazon S3 on Outposts potrebbe restituire. Per informazioni generali sugli errori di Amazon S3 e un elenco di codici di errore, consulta.

[Risposte agli errori](#)

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
BadRequest	Il bucket è in uno stato di transizione a causa di un precedente tentativo di eliminazione. Riprova più tardi.	400 Richiesta non valida	Non supportata
InvalidRequest	Questo errore può verificarsi per i seguenti motivi: • È richiesta la configurazione di Amazon VPC. • L'accesso pubblico non è consentito su S3 sui punti di accesso Outposts.	400 Richiesta non valida	Client
InvalidOutpostState	La richiesta non è valida per lo stato attuale dell'Outpost.	409 Conflitto	Non supportata

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
InvalidRequest	Il punto di accesso non si trova in uno stato in cui può essere eliminato.	400 Richiesta non valida	Non supportata
NoSuchOutpost	L'Outpost specificato non esiste.	404 Not Found (404 Non trovato)	Non supportata
UnsupportedOperation	L'azione specificata non è supportata.	404 Not Found (404 Non trovato)	Non supportata
InsufficientCapacity	Capacità insufficiente.	507 Spazio di archiviazione insufficiente	Non supportata

Elenco dei codici di errore di Amazon S3 Storage Lens

La tabella seguente contiene errori speciali che le operazioni di Amazon S3 Storage Lens potrebbero restituire. Per informazioni generali sugli errori generali di Amazon S3 e un elenco di codici di errore, consulta. [Risposte agli errori](#)

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
AccessDenied	Questa regione non è supportata come regione principale per S3 Storage Lens.	403 Forbidden	Non supportata

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
AccountNotAuthorized	Questo account non è autorizzato all'uso AWS Organizations. Usa il tuo account di gestione o l'account amministratore delegato.	403 Forbidden	Non supportata
ActivityMetricsMustEnabled	Le metriche delle attività devono essere abilitate.	400 Richiesta non valida	Non supportata
AWSOrganizationsNotInUseException	Questo account non fa parte della tua organizzazione.	403 Forbidden	Non supportata
DefaultConfigurationDeleteForbidden	La configurazione predefinita non può essere eliminata.	403 Forbidden	Non supportata
DuplicateStorageLensGroupARN	In questa configurazione sono presenti due o più voci dello stesso ARN del gruppo Storage Lens.	400 Bad Request	Non supportata
EmptyExcludeContainer	Questo errore si verifica per i seguenti motivi: • Il contenitore di esclusione non può essere vuoto. • Il contenitore di esclusione non può avere zero bucket. • Il contenitore di esclusione non può avere zero regioni.	400 Richiesta non valida	Non supportata

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
EmptyExcludeElement	È necessario specificare un gruppo Storage Lens con l'elemento Exclude.	400 Bad Request	Non supportata
EmptyIncludeContainer	Questo errore si verifica per i seguenti motivi: • Il contenitore include non può essere vuoto. • Il contenitore di inclusione non può avere zero bucket. • Il contenitore di inclusione non può avere zero regioni.	400 Richiesta non valida	Non supportata
InvalidAWSOrgArn	C'è un AWS Organizations ARN non valido nella configurazione.	400 Richiesta non valida	Non supportata
EmptyIncludeElement	È necessario specificare un gruppo Storage Lens con l'elemento Include.	400 Bad Request	Non supportata
InvalidBucketFilter	Organization-level le configurazioni non supportano i filtri bucket.	400 Richiesta non valida	Non supportata
InvalidConfigId	L'ID di configurazione non è valido.	400 Richiesta non valida	Non supportata

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
InvalidDestination	L'ARN del bucket S3 non è valido.	400 Richiesta non valida	Non supportata
InvalidEncryptionMethod	È possibile specificare un solo metodo di crittografia.	400 Richiesta non valida	Non supportata
InvalidFilterForDefaultConfiguration	La configurazione predefinita non deve includere alcun filtro.	400 Richiesta non valida	Non supportata
InvalidIncludeExcludeContainers	È possibile specificare un contenitore Include o un contenitore Exclude in una configurazione. Non è possibile specificarli entrambi in una configurazione.	400 Richiesta non valida	Non supportata
InvalidIncludeExcludeElements	È consentito un solo elemento Include o Exclude. Deve essere presente almeno un elemento Include o Exclude.	400 Bad Request	Non supportata
InvalidKMSEncryptionKeyId	L'ID ARN della chiave KMS non è valido.	400 Richiesta non valida	Non supportata
InvalidMaximumPrefixDepth	MaxDepth deve essere compreso nell'intervallo [1,10].	400 Richiesta non valida	Non supportata

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
InvalidMinimumStorageBytesPercentage	MinStorageBytesPercentage deve essere compreso nell'intervallo [1,00,100.00].	400 Richiesta non valida	Non supportata
InvalidOrganizationARN	L' AWS Organizations ARN nella configurazione non è valido.	400 Richiesta non valida	Non supportata
InvalidOrganizationForDefaultConfiguration	La configurazione predefinita non supporta le metriche a livello di organizzazione.	400 Richiesta non valida	Non supportata
InvalidRegionForDefaultConfiguration	La regione specificata non è supportata per la configurazione predefinita.	400 Richiesta non valida	Non supportata
InvalidRegionName	Il nome della regione non è valido.	400 Richiesta non valida	Non supportata
InvalidStorageLensArn	L'ARN S3 Storage Lens non è richiesto nell'input.	400 Richiesta non valida	Non supportata

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
InvalidStorageLensGroupARN	Questo ARN del gruppo Storage Lens non è valido o sono consentiti solo i gruppi Storage Lens nel tuo account. Inoltre, devi seguire la struttura ARN del gruppo Storage Lens: <code>arn::s3:::storage-lens-group/</code> e rispettare il limite di 64 caratteri. I nomi dei gruppi Storage Lens possono inoltre contenere solo i seguenti caratteri: a-z, 0-9 A-Z, trattini (-) e trattini bassi (_).	400 Bad Request	Non supportata
MissingAccountLevelActivityMetrics	Le metriche delle attività devono essere abilitate a livello di account quando le metriche di attività sono abilitate a livello di bucket.	400 Richiesta non valida	Non supportata
MissingBucketLevelActivityMetrics	Le metriche delle attività devono essere abilitate a livello di bucket quando le metriche di attività sono abilitate a livello di account.	400 Richiesta non valida	Non supportata
MissingEncryptionMethod	Il metodo di crittografia non può essere vuoto. Specifica SSE-KMS o SSE-S3.	400 Richiesta non valida	Non supportata
MissingPrefixLevelStorageMetrics	Le metriche di archiviazione a livello di prefisso sono obbligatorie quando il livello di prefisso è abilitato.	400 Richiesta non valida	Non supportata

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
OrganizationAccess Denied	Questo account non è autorizzato all'aggiunta. AWS Organizations	403 Non consentito	Non supportata
OrgConfigurationNotSupported	La regione specificata non è AWS Organizations supportata nella configurazione.	403 Non consentito	Non supportata
ServiceNotEnabledForOrg	Il ruolo collegato al servizio S3 Storage Lens non è abilitato per l'organizzazione.	403 Non consentito	Non supportata
StorageMetricsMustEnabled	Prefix-level le metriche di archiviazione devono essere abilitate.	400 Richiesta non valida	Non supportata
TooManyBuckets	Il contenitore dei secchi non può contenere più di 50 secchi.	400 Richiesta non valida	Non supportata
TooManyRegions	Il contenitore Regioni non può contenere più di 50 regioni.	400 Richiesta non valida	Non supportata
TooManyStorageLens Groups	Non puoi allegare più di 50 gruppi Storage Lens alla dashboard di Storage Lens.	400 Bad Request	Non supportata

La tabella seguente contiene errori speciali che le operazioni dei gruppi di S3 Storage Lens potrebbero restituire. Per informazioni generali sugli errori generali di Amazon S3 e un elenco di codici di errore, consulta. [Risposte agli errori](#)

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
AccessDenied	Non sei autorizzato a eseguire azioni di gruppo Storage Lens. Questa regione non è supportata come regione principale per i gruppi S3 Storage Lens.	403 Forbidden	Non supportata
ConfigurationAlreadyExists	La configurazione specificata esiste già.	409 Conflict	Non supportata
DuplicateElement	I tag devono essere univoci. L'operatore And logico include chiavi di tag duplicate. L'operatore Or logico include tag duplicati. L'operatore logico include prefissi o suffissi duplicati.	400 Bad Request	Non supportata
InvalidAge	DaysLessThan e DaysGreaterThan devono essere numeri positivi.	400 Bad Request	Non supportata
InvalidFilter	Un filtro deve includere uno dei seguenti elementi: And,Or,MatchAnyTag,MatchAnyPrefix,MatchAnySuffix,MatchObjectAge,MatchObjectSize.	400 Bad Request	Non supportata
InvalidLogicalOperator	Almeno due elementi secondari devono essere presenti negli operatori logici And o Or.	400 Bad Request	Non supportata

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
InvalidMatchAnyPrefix	Il MatchAnyPrefix parametro non può essere vuoto.	400 Bad Request	Non supportata
InvalidMatchAnySuffix	Il MatchAnySuffix parametro non può essere vuoto.	400 Bad Request	Non supportata
InvalidMatchAnyTag	Il MatchAnyTag parametro non può essere vuoto.	400 Bad Request	Non supportata
InvalidMatchObjectAge	Il MatchObjectAge parametro non può essere vuoto.	400 Bad Request	Non supportata
InvalidMatchObjectSize	Il MatchObjectSize parametro non può essere vuoto.	400 Bad Request	Non supportata
InvalidName	Il parametro Storage Lens group Name deve contenere da 1 a 64 caratteri. Il Name parametro di gruppo Storage Lens deve utilizzare il <code>^[a-zA-Z0-9\-_]+\$</code> pattern.	400 Bad Request	Non supportata
InvalidNumericCombination	Questa combinazione di età o dimensione dell'oggetto non è valida.	400 Bad Request	Non supportata
InvalidPrefix	La lunghezza massima di un prefisso è di 1.024 caratteri. La stringa del prefisso non può essere vuota.	400 Bad Request	Non supportata

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
InvalidSize	BytesLessThan e BytesGreaterThan devono essere numeri positivi. La dimensione massima dell'oggetto non può superare i 50 TB. La dimensione minima dell'oggetto non può essere maggiore o uguale a 50 TB.	400 Bad Request	Non supportata
InvalidSuffix	La lunghezza massima di un suffisso è di 1.024 caratteri. La stringa del suffisso non può essere vuota.	400 Bad Request	Non supportata
InvalidTag	La chiave del tag dell'oggetto non può superare i 128 caratteri. La stringa chiave del tag oggetto non può essere nulla o vuota. La lunghezza massima di un valore di tag è 256 caratteri. La chiave del tag object contiene caratteri non validi. La chiave del tag object deve contenere solo a-z A-Z, 0-9, spazi e i seguenti caratteri: <code>^(_\. :/+=\ -@]*)*\$</code>	400 Bad Request	Non supportata
MismatchedName	Il nome specificato nella richiesta non corrisponde al nome del gruppo Storage Lens.	400 Bad Request	Non supportata
TooManyConfigurations	Hai tentato di creare più configurazioni di gruppo Storage Lens rispetto alle 50 consentite.	400 Bad Request	Non supportata

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
TooManyElements	ElementSupera il numero massimo di elementi consentiti all'interno di un operatore logico. Sono consentiti solo 10 prefissi, suffissi o tag.	400 Bad Request	Non supportata

Elenco dei codici di errore di Amazon S3 Object Lambda

La tabella seguente contiene errori speciali che S3 Object Lambda potrebbe restituire. Per informazioni sugli errori generali di Amazon S3 e un elenco di codici di errore, consulta [Risposte agli errori](#)

Le risposte di errore ricevute dai punti di accesso di supporto durante le mancate GetObject richieste vengono inviate al chiamante inalterate.

Codice di errore	Description	Codice di stato HTTP	
LambdaInvalidResponse	Restituito al chiamante originale quando WriteGetObjectResponse risponde con a. ValidationError AWS Lambda Vedi il ValidationError messaggio per maggiori dettagli. Non tutti i casi ValidationError generano un LambdaInvalidResponse errore.	400 Bad Request	
LambdaInvocationFailed	Invocazione della funzione Lambda non riuscita.	400 Bad Request	

Codice di errore	Description	Codice di stato HTTP	
	I chiamanti potrebbero ricevere il seguente errore quando Lambda per oggetti S3 non è in grado di invocare correttamente la funzione Lambda configurata. Il messaggio di errore può contenere dettagli su un eventuale errore restituito dal AWS Lambda servizio quando si richiama la funzione (ad esempio, codice di stato, codice di errore, messaggio di errore e ID richiesta).		
LambdaNotFound	La AWS Lambda funzione non è stata trovata. La funzione Lambda, la versione o l'alias configurati non sono stati trovati durante il tentativo di invocarli . Assicurarsi che la configurazione del punto di accesso Lambda per oggetti S3 punti all'ARN della funzione Lambda corretto. Il messaggio di errore può contenere dettagli su un eventuale errore restituito dal AWS Lambda servizio quando si richiama la funzione (ad esempio, codice di stato, codice di errore, messaggio di errore e ID richiesta).	404 Not Found	

Codice di errore	Description	Codice di stato HTTP	
LambdaPermissionError	Il chiamante non è autorizzato a invocare la funzione Lambda. Il chiamante deve disporre dell'autorizzazione per invocare la funzione Lambda. Controllare le policy collegate al chiamante e assicurarsi che disponga dell'autorizzazione per utilizzare <code>lambda:Invoke</code> per la funzione configurata. Il messaggio di errore può contenere dettagli su un eventuale errore restituito dal AWS Lambda servizio quando si richiama la funzione (ad esempio, codice di stato, codice di errore, messaggio di errore e ID richiesta).	403 Forbidden	
LambdaResponseNotReceived	La funzione Lambda è terminata senza chiamare correttamente <code>WriteGetObjectResponse</code>. <code>GetObject</code> i dati di risposta vengono forniti dalla funzione Lambda richiamando l'API <code>WriteGetObjectResponse</code> operazione API. CloudWatch log di Amazon per la funzione potrebbero fornire maggiori informazioni sul motivo per cui la funzione non ha chiamato correttamente questa operazione API nonostante sia uscita normalmente.	500 InternalServerError	

Codice di errore	Description	Codice di stato HTTP	
LambdaRuntimeError	La funzione Lambda non è riuscita durante l'esecuzione. È stato ricevuto un errore esplicito dalla funzione Lambda. Per informazioni dettagliate sull'errore, consulta i AWS CloudFormation log.	500 Internal Service Error	
LambdaTimeout	La funzione Lambda non ha risposto nel tempo consentito. La funzione Lambda non è riuscita a completare la chiamata WriteGetObjectResponse entro 60 secondi.	500 Internal Service Error	

Codice di errore	Description	Codice di stato HTTP	
SlowDown	Riduci il tasso di richieste per le operazioni che comportano AWS Lambda. L'invocazione della funzione è stata rallentata AWS Lambda, forse perché ha raggiunto il limite di concorrenza configurato. Per ulteriori informazioni, consulta Gestire la concorrenza per una funzione Lambda nella Guida per gli sviluppatori. AWS Lambda Il messaggio di errore potrebbe contenere dettagli su un eventuale errore restituito dal AWS Lambda servizio quando si richiama la funzione (ad esempio, codice di stato, codice di errore, messaggio di errore e ID richiesta).	503 Slow Down	
ValidationError	Gli errori di convalida possono essere restituiti dall'operazione WriteGetObjectResponse API e possono verificarsi per numerose ragioni. Per ulteriori dettagli, vedere il messaggio di errore.	400 Bad Request	

Elenco dei codici di errore asincroni di Amazon S3

La tabella seguente contiene errori speciali che le richieste asincrone potrebbero restituire. Per informazioni generali sugli errori di Amazon S3 e un elenco di codici di errore, consulta. [Risposte agli errori](#)

Questi errori vengono restituiti quando si chiede informazioni sullo stato di una richiesta asincrona, ad esempio utilizzando `DescribeMultiRegionAccessPointOperation`. Poiché queste richieste sono asincrone, tutti questi errori hanno un codice di stato pari a `200 OK`.

Codice di errore	Description	Codice di stato HTTP
<code>AccessDenied</code>	Accesso negato.	200 OK
<code>InternalErrors</code>	Si è verificato un errore interno del server.	200 OK
<code>MalformedPolicy</code>	La sintassi della politica specificata non è valida.	200 OK
<code>MultiRegionAccessPointAlreadyOwnedByYou</code>	Hai già un Multi-Region Access Point con lo stesso nome.	200 OK
<code>MultiRegionAccessPointModifiedByAnotherRequest</code>	L'azione non è riuscita perché un'altra richiesta sta modificando la risorsa specificata. Prova a inviare nuovamente la richiesta dopo che la richiesta precedente è stata completata.	200 OK
<code>MultiRegionAccessPointNotReady</code>	L' Multi-Region Access Point specificato non è pronto per essere aggiornato.	200 OK
<code>MultiRegionAccessPointSameBucketRegion</code>	I bucket utilizzati per creare un Multi-Region Access Point non possono trovarsi nella stessa regione.	200 OK
<code>MultiRegionAccessPointUnsupportedRegion</code>	Uno dei bucket forniti per creare l' Multi-Region Access Point si trova in una regione non supportata.	200 OK
<code>NoSuchBucket</code>	Il bucket specificato non esiste.	200 OK

Codice di errore	Description	Codice di stato HTTP	
NoSuchMultiRegionAccessPoint	L' Multi-Region Access Point specificato non esiste.	200 OK	

Elenco dei codici di errore di Amazon S3 Access Grants

La seguente tabella contiene errori speciali che le richieste di S3 Access Grants potrebbero restituire. Per informazioni generali sugli errori di Amazon S3 e un elenco di codici di errore, consulta. [Risposte agli errori](#)

Codice di errore	Description	Codice di stato HTTP	
AccessGrantAlreadyExists	La concessione di accesso specificata esiste già	409	
AccessGrantsInstanceAlreadyExists	L'istanza Access Grants esiste già	409	
AccessGrantsInstanceNotEmptyError	Pulisci le posizioni prima di eliminare l'istanza di Access Grants	400	
AccessGrantsInstanceDoesNotExistError	L'istanza Access Grants non esiste	404	
AccessGrantsInstanceResourcePolicyDoesNotExist	La politica sulle risorse dell'istanza Access Grants non esiste	404	
AccessGrantsLocationAlreadyExistsError	L'ubicazione specificata delle autorizzazioni di accesso esiste già	409	
AccessGrantsLocationNotEmptyError	Pulisci la posizione delle autorizzazioni di accesso prima di eliminare	400	

Codice di errore	Description	Codice di stato HTTP	
	l'ubicazione delle concessioni di accesso		
AccessGrantsLocationsQuotaExceededError	La quota di sede delle autorizzazioni di accesso è stata superata. Quota delle sedi di Access Grants:.. <i><value></i> Contatta S3 se è necessario un aumento.	409	
AccessGrantsQuotaExceededError	La quota delle autorizzazioni di accesso è stata superata. Quota per le sovvenzioni di accesso:.. <i><value></i> Contatta S3 se è necessario un aumento.	409	
InvalidTag	Nella tua richiesta sono presenti chiavi di tag duplicate. Rimuovi le chiavi tag duplicate e riprova.	400	
InvalidAccessGrant	L'Access Grant specificato non è valido	400	
InvalidAccessGrantLocation	La sede di Access Grants specificata non è valida	400	
InvalidIamRole	Il ruolo IAM specificato non è valido	400	
InvalidIdentityCenterInstance	L'istanza del centro di identità specificata non è valida	400	
InvalidResourcePolicy	La politica delle risorse specificata non è valida	400	
InvalidResourcePolicy	La politica delle risorse specificata non è valida	400	

Codice di errore	Description	Codice di stato HTTP	
InvalidTag	Questa richiesta contiene una chiave o un valore di tag non valido. I caratteri validi includono quanto segue: [a-z A-Z +-=._:/]. Le chiavi dei tag possono contenere fino a 128 caratteri. I valori dei tag possono contenere fino a 256 caratteri.	400	
NoSuchAccessGrantError	La concessione di accesso specificata non esiste	404	
NoSuchAccessGrantsLocationError	L'ubicazione delle autorizzazioni di accesso specificata non esiste	404	
AccessDenied	Non disponi <i><requested permission></i> delle autorizzazioni per il prefisso S3 richiesto: <i><requested target></i>	403 Forbidden	
StsNotAuthorizedError	Si è verificato un errore (StsNotAuthorizedError) durante la chiamata all' GetDataAccess operazione: L'utente: non access-grants.s3.amazonaws.com è autorizzato a eseguire: sulla risorsa: sts:AssumeRole <i><IAM Role ARN></i>	403	
StsPackedPolicyTooLargeError	Si è verificato un errore (StsPackedPolicyTooLargeError) durante la chiamata all' GetDataAccess operazione: token serializzato troppo grande per la sessione	400	

Codice di errore	Description	Codice di stato HTTP	
StsValidationError	Il messaggio di errore varia a seconda dell'errore di convalida.	400	
InvalidTags	Le chiavi dei tag non possono iniziare con un prefisso AWS riservato per i tag di sistema.»	400	
TooManyTags	Il numero di tag supera il limite di 50 tag. Rimuovi alcuni tag e riprova.	400	

Elenco dei codici di FSx-related errore Amazon

La tabella seguente contiene errori speciali che potrebbero restituire le richieste fatte ad Amazon FSx tramite un punto di accesso Amazon S3. Per informazioni generali sugli errori di Amazon S3 e un elenco di codici di errore, consulta. [Risposte agli errori](#)

Codice di errore	Description	Codice di stato HTTP	
AccessDenied	Questo errore può verificarsi per i seguenti motivi: - Il chiamante non è autorizzato a eseguire un'operazione S3 specificata. - Il chiamante non è autorizzato ad accedere alla risorsa Amazon FSx per la chiave specificata.	403 Forbidden	
InsufficientCapacity	È stata raggiunta la capacità massima di archiviazione del file system.	507 Insuffici	

Codice di errore	Description	Codice di stato HTTP	
		ent Capacity	
InvalidKey	La chiave specificata non è valida.	400 Bad Request	
SlowDown	Riduci il tasso di richieste.	503 Slow Down	

Elenco dei codici di errore di Amazon S3 Tables

La seguente tabella contiene errori speciali che le operazioni di Amazon S3 Tables potrebbero restituire. Per informazioni generali sugli errori di Amazon S3 e un elenco di codici di errore, consulta.

[Risposte agli errori](#)

Codice di errore	Description	Codice di stato HTTP
AccessDeniedException	L'azione non può essere eseguita perché non disponi dell'autorizzazione richiesta.	403 Forbidden
BadRequestException	La richiesta non è valida o non è valida.	400 Bad Request
ConflictException	La richiesta non è riuscita perché c'è un conflitto con una scrittura precedente. Puoi ritentare la richiesta.	409 Conflict
ForbiddenException	Il chiamante non è autorizzato a effettuare la richiesta.	403 Forbidden
InternalServerErrorException	La richiesta non è riuscita a causa di un errore interno del server.	500 Internal Server Error

Codice di errore	Description	Codice di stato HTTP
NotFoundException	La richiesta è stata rifiutata perché non è stato possibile trovare la risorsa specificata.	404 Not Found
TooManyRequestsException	Il limite del numero di richieste al secondo è stato superato.	429 Too Many Requests

Best Practice per gli errori Amazon S3

Molte risposte di errore contengono ulteriori dati strutturati destinati a essere letti e compresi da uno sviluppatore che tenti di diagnosticare gli errori di programmazione. Ad esempio, se invii un' Content-MD5 intestazione con una richiesta REST PUT che non corrisponde al digest calcolato sul server, ricevi un errore. BadDigest La risposta all'errore include anche come elementi di dettaglio il digest calcolato e il digest fornito in base alle aspettative. Durante lo sviluppo è possibile utilizzare queste informazioni per la diagnostica dell'errore. In produzione, un programma con un comportamento corretto probabilmente includerebbe queste informazioni in un registro di errore.

Quando si progetta un'applicazione da utilizzare con Amazon S3, è importante gestire correttamente gli errori Amazon S3. Questa sezione descrive i problemi da tenere in considerazione durante la progettazione dell'applicazione.

Riprova InternalErrors

Gli errori interni si verificano nell'ambiente Amazon S3.

Le richieste che ricevono una InternalError risposta potrebbero non essere state elaborate. Ad esempio, se viene restituita una richiesta PUT InternalError, un GET successivo potrebbe recuperare il valore precedente o il valore aggiornato.

Se Amazon S3 restituisce una InternalError risposta, riprova la richiesta.

Ottimizza l'applicazione in caso di errori ripetuti SlowDown

Come ogni sistema distribuito, S3 dispone di meccanismi di protezione che rilevano il consumo eccessivo di risorse, intenzionale o non intenzionale, e reagiscono di conseguenza. SlowDown

possono verificarsi errori quando un elevato tasso di richieste attiva uno di questi meccanismi. La riduzione del tasso delle richieste porterà alla diminuzione o eliminazione degli errori di questo tipo. In generale, la maggior parte degli utenti non riscontra questi errori regolarmente; tuttavia, se desideri maggiori informazioni o riscontri SlowDown errori elevati o imprevisti, pubblica un messaggio nel nostro forum per sviluppatori di [Amazon S3](#) o iscriviti a. Supporto <https://aws.amazon.com/premiumsupport/>

Errori isolati

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

Amazon S3 fornisce un insieme di codici di errori usati sia dall'API SOAP che dall'API REST. L'API SOAP restituisce codici di errore Amazon S3 standard. L'API REST è progettata per somigliare a un server HTTP standard e interagisce con i client HTTP esistenti (browser, librerie del client HTTP, proxy, cache e così via). Per garantire che i client HTTP gestiscano correttamente gli errori, Amazon S3 associa ogni errore di Amazon S3 a un codice di stato HTTP.

I codici di stato HTTP sono meno intuitivi dei codici di errore Amazon S3 e contengono un minor numero di informazioni sull'errore. Ad esempio, gli errori di Amazon S3 `NoSuchKey` e `NoSuchBucket` sono mappati entrambi sul codice di stato HTTP `404 Not Found`.

Sebbene i codici di stato HTTP contengano un minor numero di informazioni sull'errore, i client che comprendono l'HTTP ma non l'API di Amazon S3 in genere gestiscono l'errore correttamente.

Per questo motivo, quando si gestiscono o si riportano errori di Amazon S3 agli utenti finali, è opportuno utilizzare il codice di errore di Amazon S3 anziché il codice di stato HTTP, perché contiene il maggior numero possibile di informazioni sull'errore. Inoltre, quando si esegue il debug dell'applicazione, è bene consulta l'elemento `<Details>` della risposta di errore XML, che può essere letto da un interlocutore umano.

AWS Glossario

Per la AWS terminologia più recente, consultate il [AWS glossario](#) nella sezione Reference. Glossario AWS

Risorse di Amazon S3

La tabella seguente elenca le risorse correlate che ti saranno utili durante l'utilizzo di questo servizio.

Risorsa	Description
Guida per l'utente di Amazon Simple Storage Service	La guida introduttiva fornisce un breve tutorial del servizio basato su un semplice caso d'uso.
Guida per l'utente di Amazon Simple Storage Service	La guida per sviluppatori descrive come eseguire attività utilizzando le operazioni di Amazon S3.
Domande frequenti tecniche su Amazon S3	Le domande frequenti coprono le 20 domande principali poste dagli sviluppatori su questo prodotto.
Note sulla versione di Amazon S3	Le note di rilascio forniscono una panoramica di alto livello della versione corrente. In particolare, vengono indicate tutte le nuove funzioni, le correzioni e tutti i problemi noti.
Strumenti per Amazon Web Services	Un punto di partenza centrale per trovare documentazione, esempi di codice, note di rilascio e altre informazioni per aiutarvi a creare applicazioni innovative con AWS SDK e strumenti.
Console di gestione AWS	La console consente di eseguire la maggior parte delle funzioni di Amazon S3 senza programmazione.
Forum di discussione	Un forum basato sulla community in cui gli sviluppatori possono discutere di domande tecniche relative ad Amazon Web Services.
Supporto Centro	La home page del supporto AWS tecnico, che include l'accesso ai forum per sviluppatori, alle domande frequenti tecniche, alla pagina sullo stato del servizio e al supporto Premium.

Risorsa	Description
Supporto	La pagina Web principale per informazioni su Supporto un canale di supporto individuale con risposta rapida per aiutarti a creare ed eseguire applicazioni su Infrastructure Services. AWS
Informazioni sul prodotto Amazon S3	La pagina Web principale che include informazioni su Amazon S3.
Contattaci	Un punto di contatto centrale per domande riguardanti AWS fatturazione, account, eventi, abusi, ecc.
Condizioni d'uso	Informazioni dettagliate sul copyright e sull'uso dei marchi Amazon.com e su altri argomenti.

Appendice

Argomenti

- [Appendice: Risposta SelectObjectContent](#)
- [Appendice: oggetto OPTIONS](#)
- [Appendice: API SOAP](#)
- [Appendice: Autenticazione delle richieste \(AWS firma \(versione 2\)\)](#)
- [Appendice: API di configurazione del ciclo di vita \(obsolete\)](#)

Appendice: Risposta SelectObjectContent

Description

L'operazione Amazon S3 Select filtra il contenuto di un oggetto Amazon S3 in base a una semplice istruzione SQL (Structured Query Language). Poiché la dimensione della risposta di questa operazione è sconosciuta, Amazon S3 Select trasmette la risposta come una serie di messaggi e include un'Transfer-Encoding intestazione con **chunked** come valore nella risposta.

Per ulteriori informazioni su Amazon S3 Select, consulta [Selection Content from Objects](#) nella Amazon Simple Storage Service User Guide.

Per ulteriori informazioni sull'utilizzo di SQL con Amazon S3 Select, consulta [SQL Reference for Amazon S3 Select e Amazon Glacier Select](#) nella Amazon Simple Storage Service User Guide.

Risposte

Una corretta operazione Amazon S3 Select restituisce un codice di stato. 200 OK

Intestazioni di risposta

Questa implementazione dell'operazione utilizza solo le intestazioni di risposta comuni alla maggior parte delle risposte. Per ulteriori informazioni, vedere [Common Response](#) Headers.

Corpo della risposta

Poiché la dimensione della risposta Amazon S3 Select è sconosciuta, Amazon S3 trasmette la risposta come una serie di messaggi e include un'Transfer-Encoding intestazione con **chunked** come valore nella risposta. L'esempio seguente mostra il formato della risposta al livello superiore:

```
<Message 1>
<Message 2>
<Message 3>
.....
<Message n>
```

Ogni messaggio è composto da due sezioni: è costituita da due sezioni: l'introduzione e i dati. La sezione del preludio è composta da 1) la lunghezza totale in byte del messaggio e 2) la lunghezza

in byte combinata di tutte le intestazioni. La sezione dati è composta da 1) le intestazioni e 2) un payload.

Ogni sezione termina con un checksum intero big-endian (CRC) a 4 byte. Amazon S3 Select utilizza CRC32 (spesso indicato come GZIP CRC32) per calcolare entrambi i CRC. Per ulteriori informazioni su CRC32, consultare la [specifica versione 4.3 del formato di file GZIP](#).

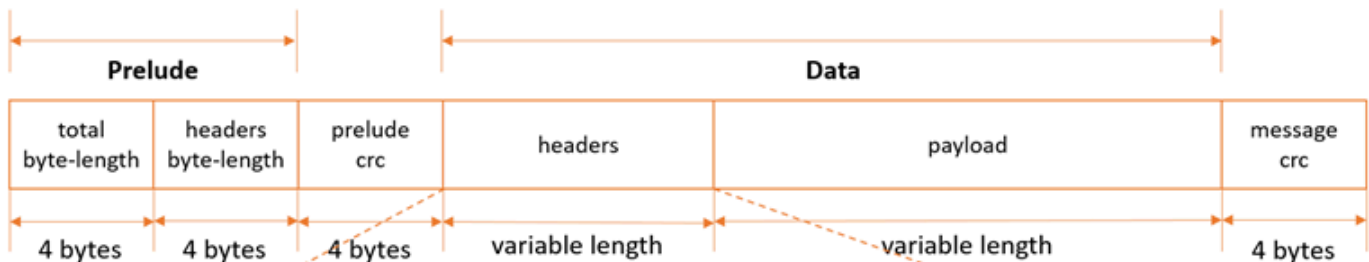
Il sovraccarico totale dei messaggi, incluso il preludio ed entrambi i checksum, è di 16 byte.

Note

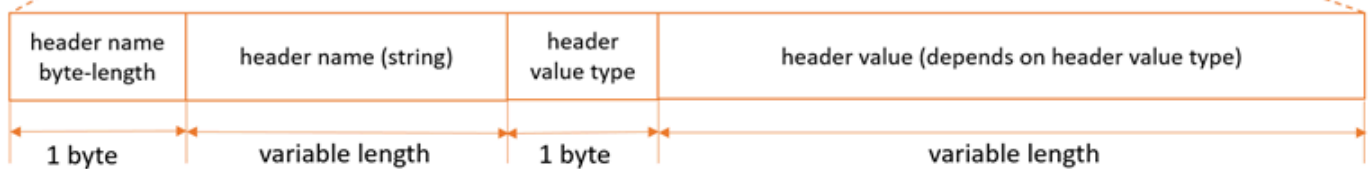
Tutti i `integer` valori all'interno dei messaggi sono in ordine di byte di rete o in ordine big-endian.

Il seguente diagramma mostra i componenti che compongono un messaggio e un'intestazione. Nota che ci sono più intestazioni per messaggio.

Message:



Headers (multiple headers per message):

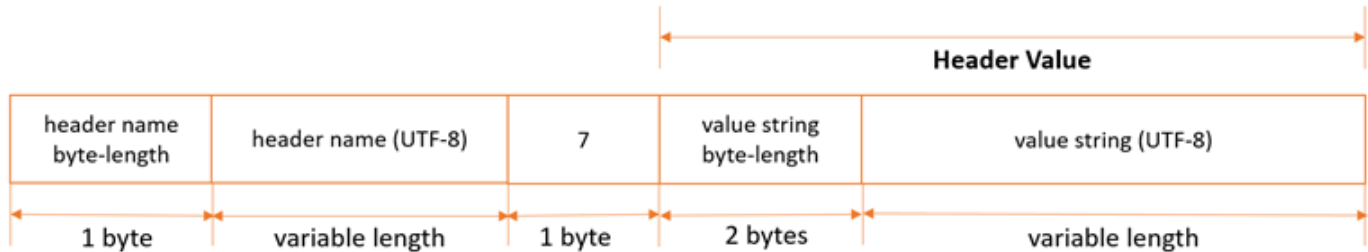


Note

Per Amazon S3 Select, il tipo di valore dell'intestazione è sempre 7 (type=String). Per questo tipo, il valore dell'intestazione è costituito da due componenti, un intero big-endian di 2 byte e

una stringa di tale lunghezza in byte. UTF-8 Il diagramma seguente mostra i componenti che compongono le intestazioni di Amazon S3 Select.

Amazon S3 Select Headers (type=String):



Calcoli della lunghezza in byte del payload (questi due calcoli sono equivalenti):

- $\text{payload\_length} = \text{total\_length} - \text{header\_length} - \text{sizeof}(\text{total\_length}) - \text{sizeof}(\text{header\_length}) - \text{sizeof}(\text{prelude\_crc}) - \text{sizeof}(\text{message\_crc})$
- $\text{payload\_length} = \text{total\_length} - \text{header\_length} - 16$

Ogni messaggio contiene i seguenti componenti:

- Preludio: dimensione sempre fissa di 8 byte (due campi da 4 byte ciascuno):
 - Primi quattro byte: lunghezza totale in byte: lunghezza in byte intera dell' Big-endian intero messaggio (incluso il campo di lunghezza totale di 4 byte stesso).
 - Secondi quattro byte: lunghezza in byte delle intestazioni: lunghezza in byte Big-endian intera della parte delle intestazioni del messaggio (escluso il campo relativo alla lunghezza delle intestazioni stesso).
- Prelude CRC: checksum intero big-endian (CRC) a 4 byte per la parte preludio del messaggio (escluso il CRC stesso). Il preludio ha un CRC separato dal CRC del messaggio (vedi sotto), per garantire che le informazioni danneggiate in byte possano essere rilevate immediatamente, senza causare un comportamento patologico di buffering.
- Intestazioni: una serie di metadati che annotano il messaggio, ad esempio il tipo di messaggio, il formato del payload e così via. I messaggi possono avere più intestazioni, quindi questa parte del messaggio può avere lunghezze di byte diverse a seconda del tipo di messaggio. Le intestazioni sono coppie chiave-valore, in cui sia la chiave che il valore sono stringhe. UTF-8 Le intestazioni

possono essere visualizzate in qualsiasi ordine all'interno della parte relativa alle intestazioni del messaggio e qualsiasi tipo di intestazione può apparire una sola volta.

Per Amazon S3 Select, di seguito è riportato un elenco di nomi di intestazione e il set di valori validi a seconda del tipo di messaggio.

- **MessageType Intestazione:**
 - HeaderName => «:message-type»
 - Valido HeaderValues => «errore», «evento»
- **EventType Intestazione:**
 - HeaderName => «:event-type»
 - Valido HeaderValues => «Record», «Cont», «Progress», «Stats», «End»
- **ErrorCode Intestazione:**
 - HeaderName => «:codice di errore»
 - Valido HeaderValues => Codice di errore dalla tabella nella sezione. [Elenco dei codici di errore SELECT Object Content](#)
- **ErrorMessage Intestazione:**
 - HeaderName => «:messaggio di errore»
 - Valido HeaderValues => Messaggio di errore restituito dal servizio, per aiutare a diagnosticare gli errori a livello di richiesta.
- **Payload:** può essere qualsiasi cosa.
- **Messaggio CRC:** checksum intero big-endian (CRC) a 4 byte dall'inizio del messaggio all'inizio del checksum (ovvero, tutto il contenuto del messaggio escluso il messaggio CRC stesso).

Ogni intestazione contiene i seguenti componenti. Possono esserci più intestazioni per messaggio.

- **Nome dell'intestazione Byte-Length:** Byte-length del nome dell'intestazione.
- **Nome dell'intestazione:** nome dell'intestazione, che indica il tipo di intestazione. Valori validi: «:message-type» «:event-type» «:error-code» «:error-message»
- **Tipo di valore dell'intestazione:** Enum che indica il tipo di valore dell'intestazione. Per Amazon S3 Select, è sempre 7.
- **Stringa di valore Byte-Length:** (per Amazon S3 Select) Byte-length della stringa del valore dell'intestazione.

- **Stringa del valore dell'intestazione:** (per Amazon S3 Select) Valore della stringa di intestazione. I valori validi per questo campo variano in base al tipo di intestazione. Consulta le sezioni seguenti per i valori validi per ogni tipo di intestazione e tipo di messaggio.

Per Amazon S3 Select, le risposte possono essere messaggi dei seguenti tipi:

- **Registra un messaggio:** può contenere un singolo record, record parziali o più record. A seconda della dimensione del risultato, una risposta può contenere uno o più di questi messaggi.
- **Messaggio di continuazione:** Amazon S3 invia periodicamente questo messaggio per mantenere aperta la connessione TCP. Questi messaggi vengono visualizzati casualmente nelle risposte. Il cliente deve rilevare il tipo di messaggio ed elaborarlo di conseguenza.
- **Messaggio di avanzamento:** Amazon S3 invia periodicamente questo messaggio, se richiesto. Contiene informazioni sullo stato di avanzamento di una query iniziata ma non ancora completata.
- **Messaggio statistico:** Amazon S3 invia questo messaggio alla fine della richiesta. Contiene statistiche sulla query.
- **Messaggio finale:** indica che la richiesta è completa e non verranno inviati altri messaggi. Non si deve dare per scontato che la richiesta sia completa finché il cliente non riceve un End messaggio.
- **RequestLevelError messaggio:** Amazon S3 invia questo messaggio se la richiesta non è riuscita per qualsiasi motivo. Contiene il codice e il messaggio di errore relativi all'errore. Se Amazon S3 invia un RequestLevelError messaggio, non invia un End messaggio.

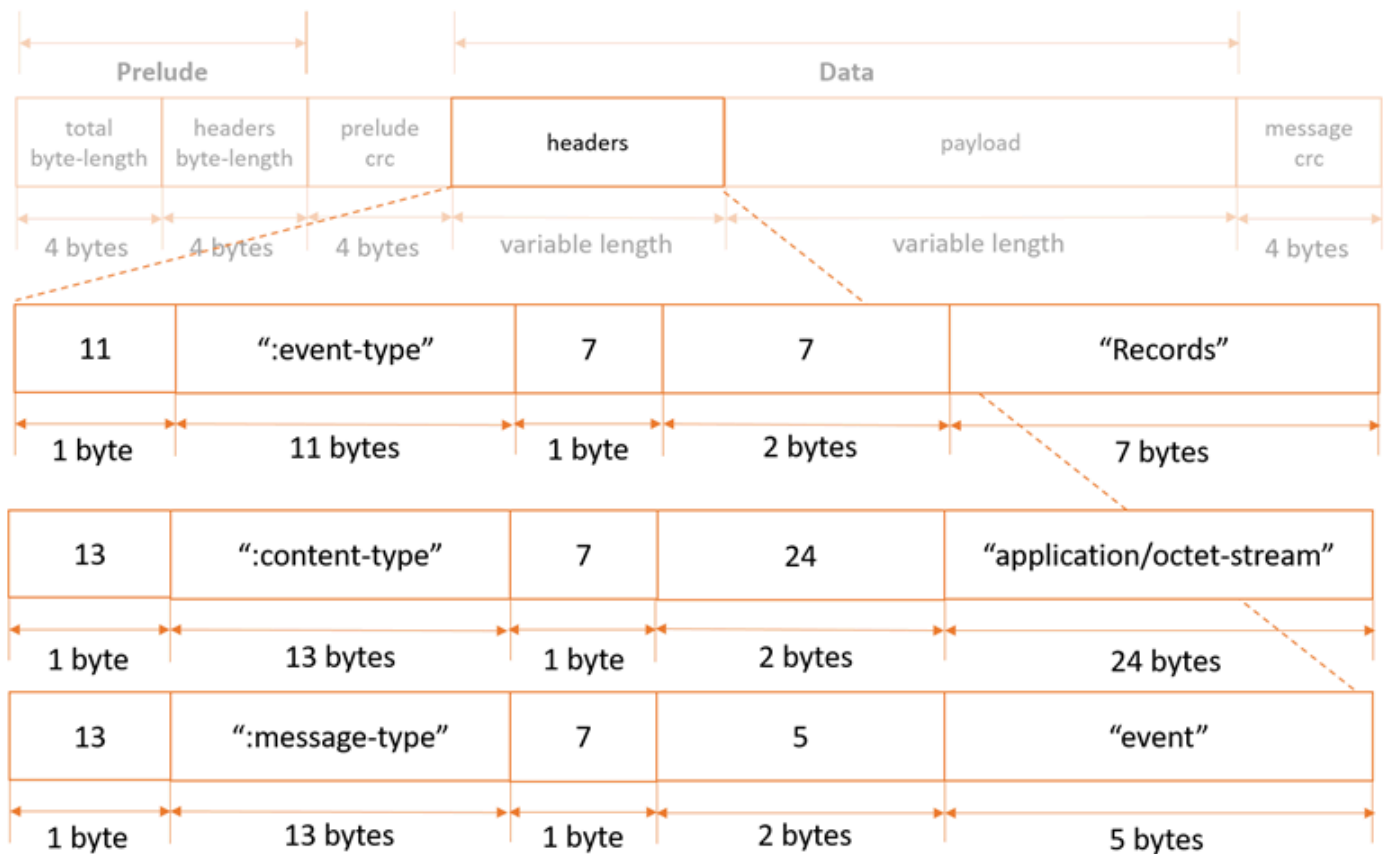
Le sezioni seguenti spiegano la struttura di ciascun tipo di messaggio in modo più dettagliato.

Per esempi di codice e test unitari che utilizzano questo protocollo, vedete [AWS C Event Stream](#) sul GitHub sito Web.

Registra il messaggio

Specifiche dell'intestazione

I messaggi dei record contengono tre intestazioni, come segue:



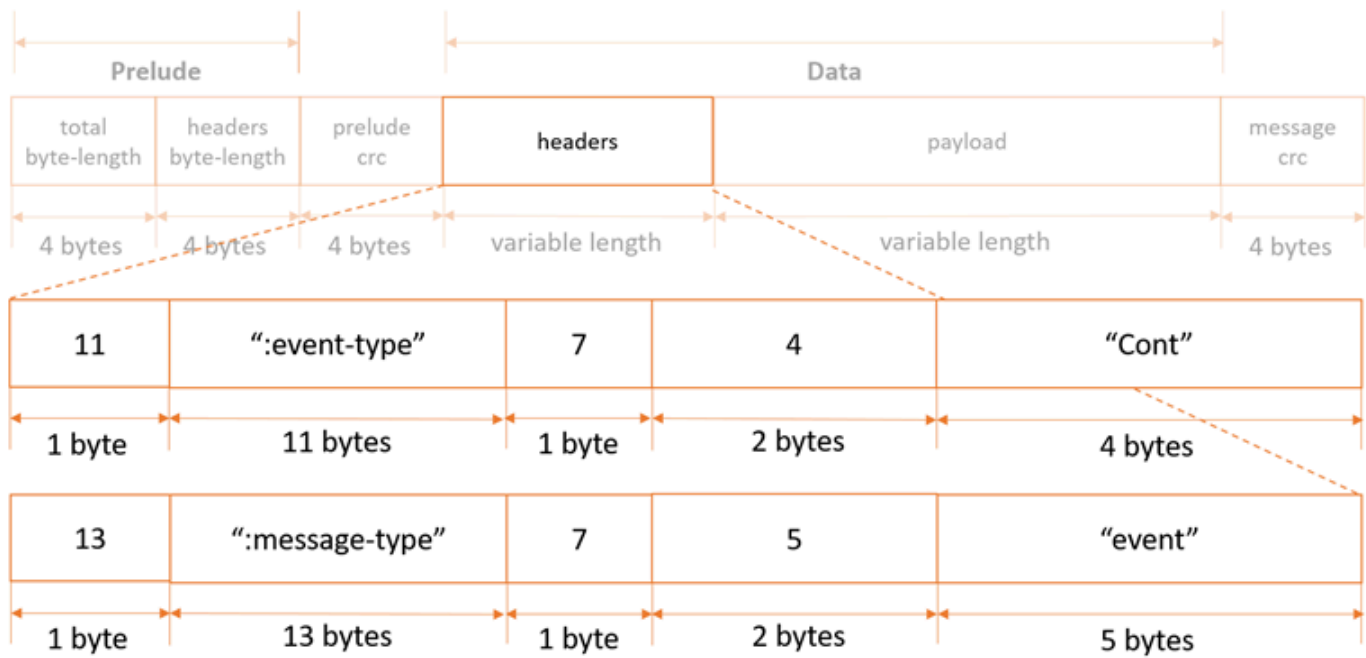
Specifiche del payload

I payload dei messaggi Records possono contenere un singolo record, record parziali o più record.

Messaggio di continuazione

Specifiche dell'intestazione

I messaggi di continuazione contengono due intestazioni, come segue:



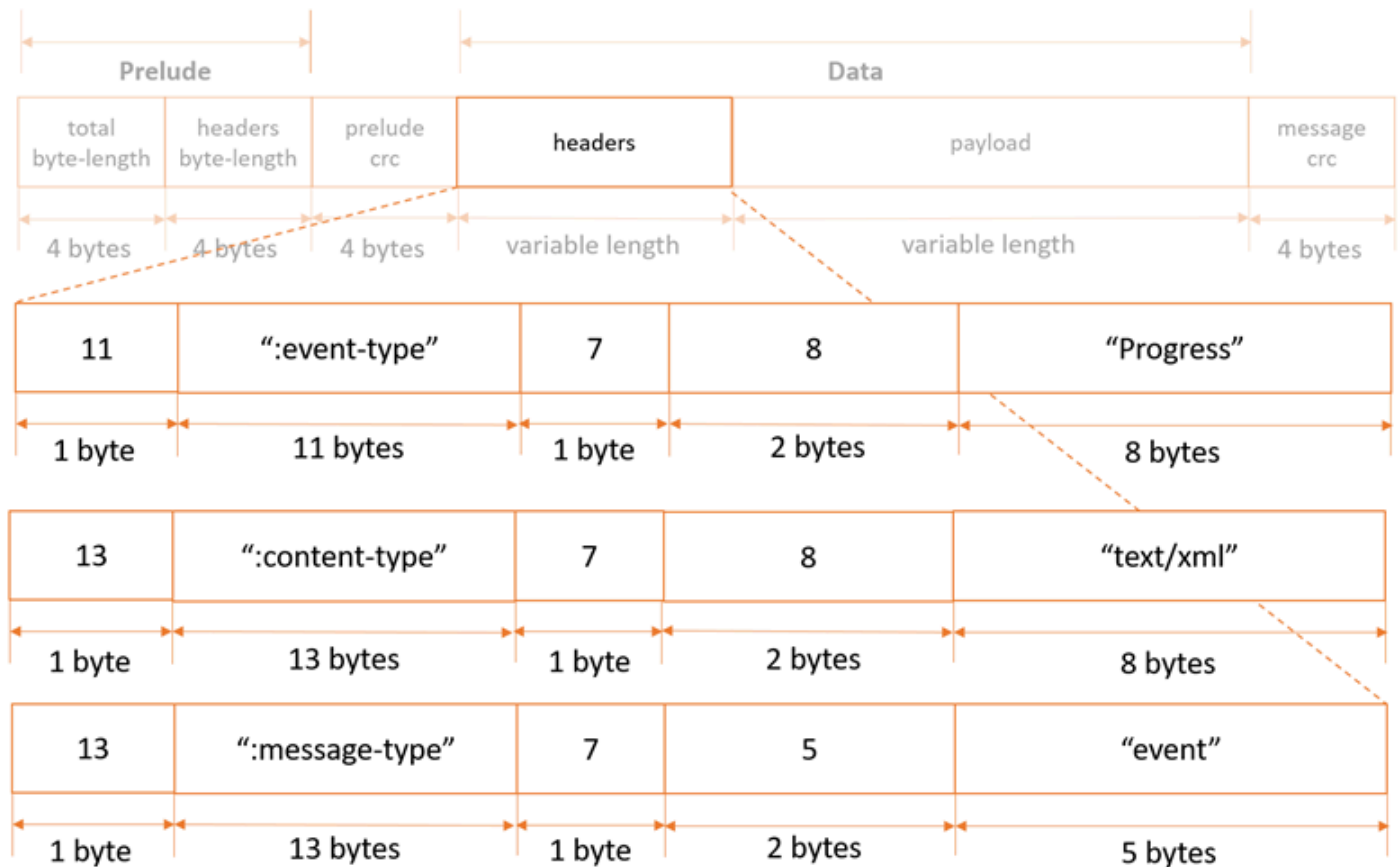
Specifiche del payload

I messaggi di continuazione non hanno alcun payload.

Messaggio di avanzamento

Specifiche dell'intestazione

I messaggi di avanzamento contengono tre intestazioni, come segue:



Specifiche del payload

Il payload dei messaggi di avanzamento è un documento XML contenente informazioni sullo stato di avanzamento di una richiesta.

- **BytesScanned**=> Numero di byte che sono stati elaborati prima di essere decompressi (se il file è compresso).
- **BytesProcessed**=> Numero di byte che sono stati elaborati dopo essere stati decompressi (se il file è compresso).
- **BytesReturned**=> Numero attuale di byte di dati di payload dei record restituiti da Amazon S3.

Per i file non compressi, e sono uguali. **BytesScanned BytesProcessed**

Esempio:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

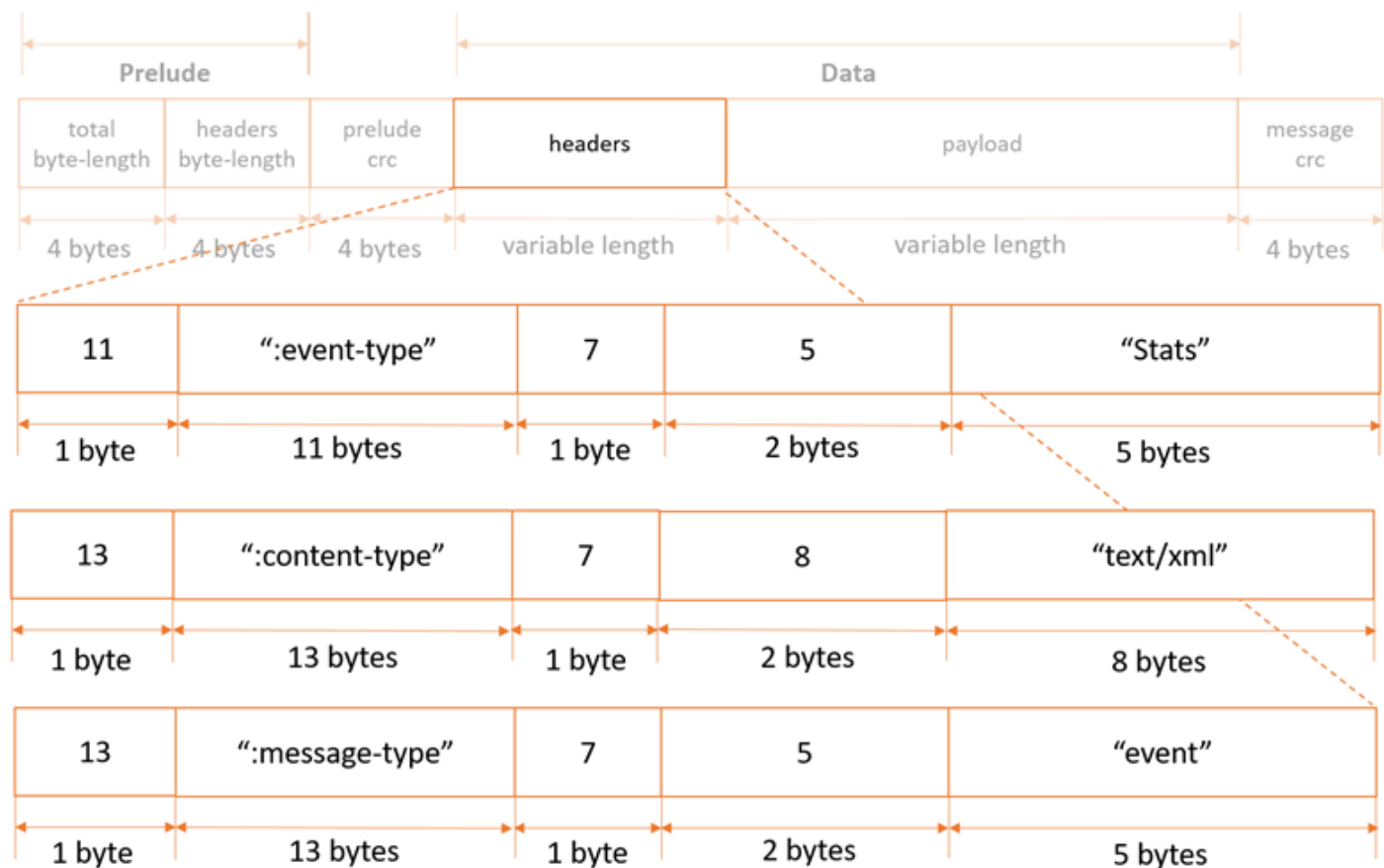
<Progress>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
</Progress>

```

Messaggio di statistiche

Specifiche dell'intestazione

I messaggi di statistica contengono tre intestazioni, come segue:



Specifiche del payload

Il payload dei messaggi di statistica è un documento XML contenente informazioni sulle statistiche di una richiesta al termine dell'elaborazione.

- BytesScanned=> Numero di byte che sono stati elaborati prima di essere decompressi (se il file è compresso).

- BytesProcessed=> Numero di byte che sono stati elaborati dopo essere stati decompressi (se il file è compresso).
- BytesReturned=> Numero totale di byte di dati di payload dei record restituiti da Amazon S3.

Per i file non compressi, e sono uguali. BytesScanned BytesProcessed

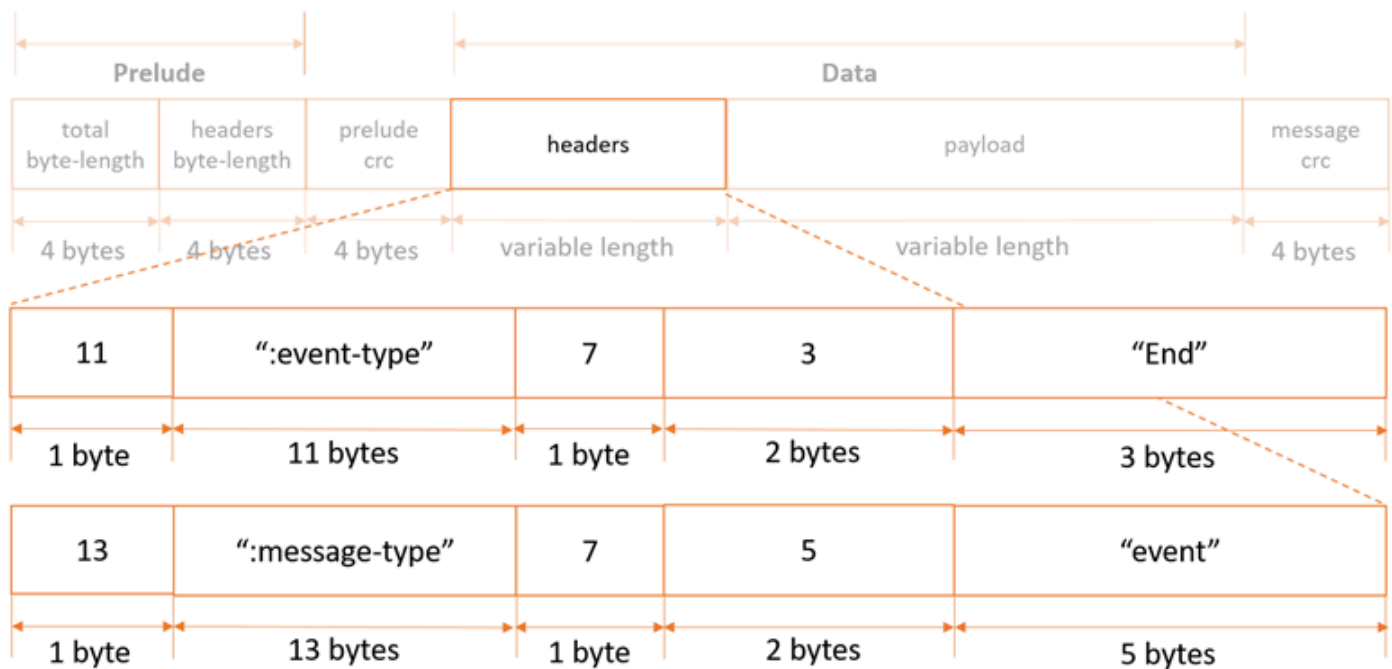
Esempio:

```
<?xml version="1.0" encoding="UTF-8"?>
<Stats>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
</Stats>
```

Fine messaggio

Specifiche dell'intestazione

I messaggi finali contengono due intestazioni, come segue:



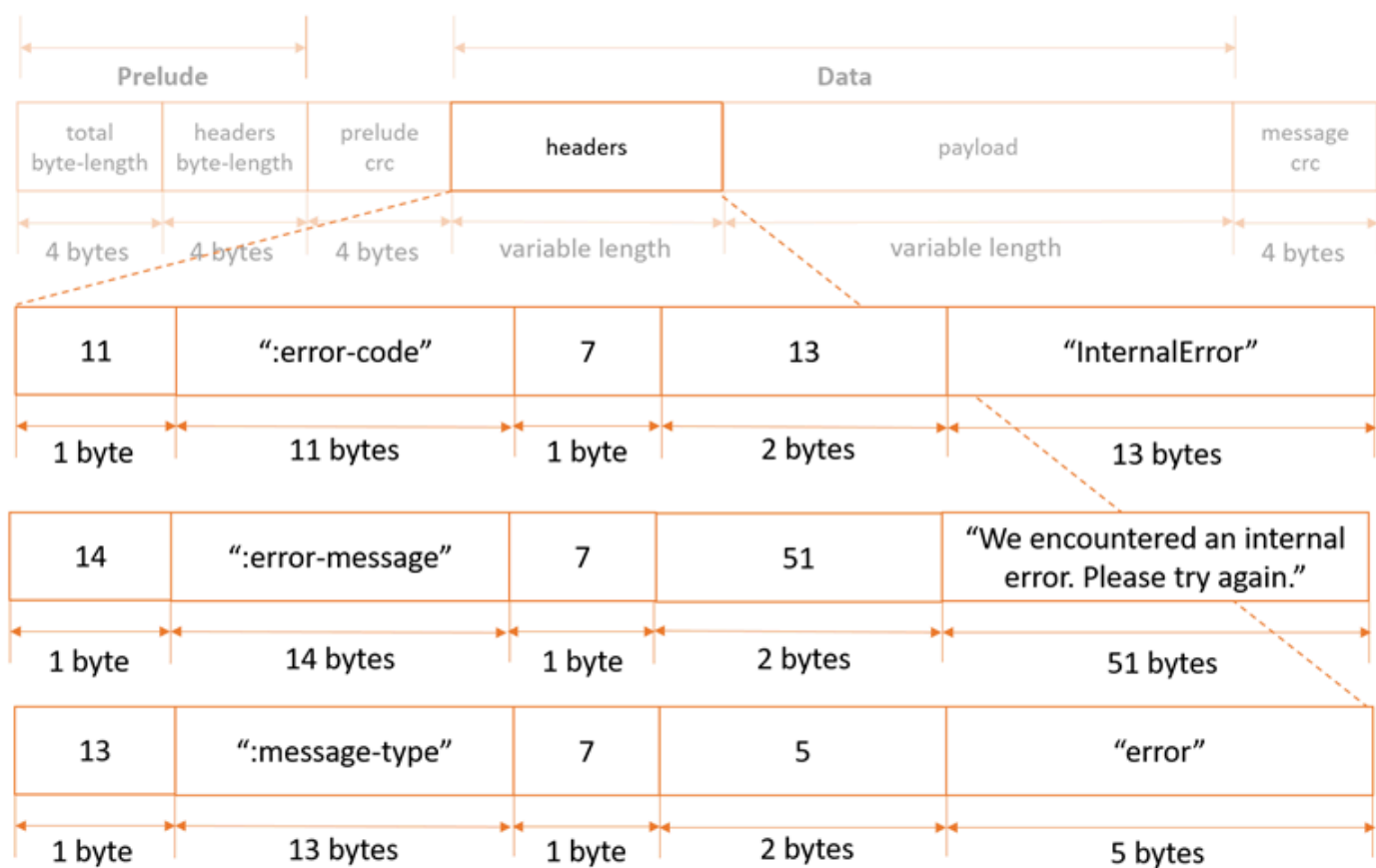
Specifiche del payload

I messaggi finali non hanno alcun payload.

Messaggio di errore relativo al livello di richiesta

Specifiche dell'intestazione

Request-level i messaggi di errore contengono tre intestazioni, come segue:



Per un elenco dei possibili codici e messaggi di errore, vedere. [Elenco dei codici di errore SELECT Object Content](#)

Specifiche del payload

Request-level i messaggi di errore non hanno alcun payload.

Risorse correlate

- [SelectObjectContent](#)

- [GetObject](#)
- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)

Appendice: oggetto OPTIONS

Description

Un browser può inviare questa richiesta di preflight ad Amazon S3 per determinare se può inviare una richiesta effettiva con l'origine, il metodo HTTP e le intestazioni specifici.

Amazon S3 supporta la condivisione di risorse tra origini (CORS) consentendo di aggiungere una sottorisorsa su un bucket. cors Quando un browser invia questa richiesta di preflight, Amazon S3 risponde valutando le regole definite nella configurazione. cors

Se non cors è abilitato nel bucket, Amazon S3 restituisce una risposta. 403 Forbidden

Per ulteriori informazioni su CORS, consulta la sezione [Enabling Cross-Origin Resource Sharing](#) nella Amazon Simple Storage Service User Guide.

Richieste

Sintassi

```
OPTIONS /ObjectName HTTP/1.1
Host: BucketName.s3.amazonaws.com
Origin: Origin
Access-Control-Request-Method: HTTPMethod
Access-Control-Request-Headers: RequestHeader
```

Parametri della richiesta

Questa operazione non introduce parametri di richiesta specifici, ma può contenere tutti i parametri di richiesta richiesti dalla richiesta effettiva.

Intestazioni di richiesta

Nome	Description	Richiesto
Origin	Identifica l'origine della richiesta tra origini diverse ad Amazon S3. Ad esempio, http://www.example.com. Tipo: stringa	Sì

Nome	Description	Richiesto
	Impostazione predefinita: nessuna	
Access-Control-Request-Method	Identifica il metodo HTTP che verrà utilizzato nella richiesta effettiva. Tipo: stringa Impostazione predefinita: nessuna	Sì
Access-Control-Request-Headers	Un elenco delimitato da virgole di intestazioni HTTP che verranno inviate nella richiesta effettiva. Ad esempio, per inserire un oggetto con crittografia lato server, questa richiesta di preflight determinerà se può includere l'intestazione nella richiesta. <code>x-amz-server-side-encryption</code> Tipo: stringa Impostazione predefinita: nessuna	No

Elementi della richiesta

Questa implementazione dell'operazione non utilizza elementi di richiesta.

Risposte

Intestazioni di risposta

Header	Description
Access-Control-Allow-Origin	L'origine che hai inviato nella tua richiesta. Se l'origine della tua richiesta non è consentita, Amazon S3 non includerà questa intestazione nella risposta.

Header	Description
	Tipo: stringa
Access-Control-Max-Age	Per quanto tempo, in secondi, i risultati della richiesta di preflight possono essere memorizzati nella cache. Tipo: stringa
Access-Control-Allow-Methods	Il metodo HTTP inviato nella richiesta originale. Se il metodo nella richiesta non è consentito, Amazon S3 non includerà questa intestazione nella risposta. Tipo: stringa
Access-Control-Allow-Headers	Un elenco delimitato da virgole di intestazioni HTTP che il browser può inviare nella richiesta effettiva. Se una delle intestazioni richieste non è consentita, Amazon S3 non includerà tale intestazione nella risposta, né la risposta conterrà nessuna delle intestazioni con il prefisso. Access-Control Tipo: stringa
Access-Control-Expose-Headers	Un elenco di intestazioni HTTP delimitato da virgole. Questa intestazione fornisce al JavaScript client l'accesso a queste intestazioni nella risposta alla richiesta effettiva. Tipo: stringa

Elementi di risposta

Questa implementazione dell'operazione non restituisce elementi di risposta.

Esempi

Esempio: invia una richiesta OPTIONS di preflight a un bucket abilitato per **cors**

Un browser può inviare questa richiesta di preflight ad Amazon S3 per determinare se può inviare la richiesta PUT effettiva dall'`http://www.example.com`origine al bucket Amazon S3 denominato `examplebucket`

Richiesta di esempio

```
OPTIONS /exampleobject HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Origin: http://www.example.com
Access-Control-Request-Method: PUT
```

Risposta di esempio

```
HTTP/1.1 200 OK
x-amz-id-2: 6SvaESv3VULYPLik5LL171SPPtSnBvDdGmnk1X1HfU17uS2m1DF6td6KWKNjYMXZ
x-amz-request-id: BDC4B83DF5096BBE
Date: Wed, 21 Aug 2012 23:09:55 GMT
Etag: "1f1a1af1f111111111111111c11aed1da1"
Access-Control-Allow-Origin: http://www.example.com
Access-Control-Allow-Methods: PUT
Access-Control-Expose-Headers: x-amz-request-id
Content-Length: 0
Server: AmazonS3
```

Risorse correlate

- [GetBucketCors](#)
- [DeleteBucketCors](#)
- [PutBucketCors](#)

Appendice: API SOAP

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

Questa sezione descrive l'API SOAP per quanto riguarda le operazioni su servizi, bucket e oggetti. Nota che le richieste SOAP, sia autenticate che anonime, devono essere inviate ad Amazon S3 tramite SSL. Amazon S3 restituisce un errore quando invii una richiesta SOAP tramite HTTP.

La versione più recente di Amazon S3 WSDL è disponibile all'indirizzo docs.aws.amazon.com/200601/AmazonS3-03-.wsdl.

Argomenti

- [Operazioni sul servizio \(API SOAP\)](#)
- [Operazioni sui bucket \(API SOAP\)](#)
- [Operazioni sugli oggetti \(API SOAP\)](#)
- [Autenticazione delle richieste SOAP](#)
- [Impostazione della policy d'accesso con SOAP](#)
- [Elementi comuni](#)
- [Risposte agli errori SOAP](#)

Operazioni sul servizio (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

Questa sezione descrive le operazioni che è possibile eseguire sul servizio Amazon S3.

Argomenti

- [ListAllMyBuckets \(API SOAP\)](#)

ListAllMyBuckets (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

L'ListAllMyBuckets operazione restituisce un elenco di tutti i bucket di proprietà del mittente della richiesta.

Example

Sample Request

```
<ListAllMyBuckets xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListAllMyBuckets>
```

Sample Response

```
<ListAllMyBucketsResult xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <Owner>
    <ID>bcaf1ffd86f41161ca5fb16fd081034f</ID>
    <DisplayName>webfile</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <Name>quotes</Name>
      <CreationDate>2006-02-03T16:45:09.000Z</CreationDate>
    </Bucket>
    <Bucket>
      <Name>samples</Name>
      <CreationDate>2006-02-03T16:41:58.000Z</CreationDate>
    </Bucket>
  </Buckets>
```

```
</ListAllMyBucketsResult>
```

Corpo della risposta

- **Owner :**

Ciò fornisce informazioni che Amazon S3 utilizza per rappresentare la tua identità ai fini dell'autenticazione e del controllo degli accessi. L'ID è un identificatore univoco e permanente per lo sviluppatore che ha effettuato la richiesta. DisplayName è un nome leggibile dall'uomo che rappresenta lo sviluppatore che ha effettuato la richiesta. Non è univoco e potrebbe cambiare nel tempo. Ti consigliamo di abbinare il nome del tuo DisplayName forum.

- **Name :**

Il nome di un secchio. Tieni presente che se uno dei tuoi bucket è stato eliminato di recente, il nome del bucket eliminato potrebbe essere ancora presente in questo elenco per un certo periodo di tempo.

- **CreationDate:**

L'ora in cui il bucket è stato creato.

Controllo degli accessi

È necessario autenticarsi con un ID chiave di AWS accesso valido. Le richieste anonime non sono mai consentite per elencare i bucket e puoi elencare solo i bucket di cui sei il proprietario.

Operazioni sui bucket (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

Questa sezione descrive le operazioni che puoi eseguire sui bucket Amazon S3.

Argomenti

- [CreateBucket \(API SOAP\)](#)
- [DeleteBucket \(API SOAP\)](#)

- [ListBucket \(API SOAP\)](#)
- [GetBucketAccessControlPolicy \(API SOAP\)](#)
- [SetBucketAccessControlPolicy \(API SOAP\)](#)
- [GetBucketLoggingStatus \(API SOAP\)](#)
- [SetBucketLoggingStatus \(API SOAP\)](#)

CreateBucket (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

L'CreateBucket operazione crea un bucket. Non tutte le stringhe sono nomi di bucket accettabili. Per informazioni sulle restrizioni relative ai nomi dei bucket, consulta [Lavorare con i bucket Amazon S3](#).

Note

Per determinare se esiste un nome di bucket, usa ListBucket e imposta su 0. MaxKeys. Una NoSuchBucket risposta indica che il bucket è disponibile, una AccessDenied risposta indica che qualcun altro è il proprietario del bucket e una risposta di successo indica che sei il proprietario del bucket o hai l'autorizzazione per accedervi.

Example Crea un bucket chiamato «quotes»

Sample Request

```
<CreateBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</CreateBucket>
```

Sample Response

```
<CreateBucketResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <CreateBucketResponse>
    <Bucket>quotes</Bucket>
  </CreateBucketResponse>
</CreateBucketResponse>
```

Elementi

- **Bucket**: Il nome del bucket che stai cercando di creare.
- **AccessControlList**: L'elenco di controllo degli accessi per il nuovo bucket. Questo elemento è facoltativo. Se non viene fornito, il bucket viene creato con una politica di accesso che consente l'accesso FULL_CONTROL al richiedente.

Controllo degli accessi

È necessario autenticarsi con un ID chiave di accesso valido. AWS Le richieste anonime non sono mai autorizzate a creare bucket.

Risorse correlate

- [ListBucket \(API SOAP\)](#)

DeleteBucket (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

L'DeleteBucket operazione elimina un bucket. Tutti gli oggetti nel bucket devono essere eliminati prima che il bucket stesso possa essere eliminato.

Example

Questo esempio elimina il bucket «quotes».

Sample Request

```
<DeleteBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId> AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</DeleteBucket>
```

Sample Response

```
<DeleteBucketResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <DeleteBucketResponse>
    <Code>204</Code>
    <Description>No Content</Description>
  </DeleteBucketResponse>
</DeleteBucketResponse>
```

Elementi

- **Bucket** : Il nome del bucket che desideri eliminare.

Controllo degli accessi

Solo il proprietario di un bucket può eliminarlo, indipendentemente dalla politica di controllo degli accessi sul bucket.

ListBucket (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

L'ListBucket operazione restituisce informazioni su alcuni elementi del bucket.

Per un'introduzione generale all'operazione di elenco, vedere [Listing Object Keys](#).

Richieste

Questo esempio elenca fino a 1000 chiavi nel bucket «quotes» che hanno il prefisso «notes».

Sintassi

```
<ListBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Prefix>notes</Prefix>
  <Delimiter></Delimiter>
  <MaxKeys>1000</MaxKeys>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListBucket>
```

Parametri

Nome	Description	Richiesto
<code>prefix</code>	Limita la risposta alle chiavi che iniziano con il prefisso indicato. È possibile utilizzare i prefissi per separare un bucket in diversi set di chiavi in modo simile a come un file system utilizza le cartelle. Tipo: stringa Impostazione predefinita: nessuna	No
<code>marker</code>	Indica da dove iniziare l'elenco nel bucket. L'elenco includerà solo le chiavi che compaiono lessicograficamente dopo il marker. Questo è utile per l'impaginazione: per ottenere la pagina successiva dei risultati, usa l'ultima chiave della pagina corrente come indicatore. Tipo: stringa Impostazione predefinita: nessuna	No
<code>max-keys</code>	Il numero massimo di tasti che vorresti vedere nel corpo della risposta. Il server potrebbe restituire meno di questo numero di chiavi, ma non ne restituirà di più. Tipo: stringa	No

Nome	Description	Richiesto
	Impostazione predefinita: nessuna	
delimiter	Fa sì che le chiavi che contengono la stessa stringa tra il prefisso e la prima occorrenza del delimitatore vengano raggruppate in un unico elemento risultato della raccolta. CommonPrefixes Queste chiavi raggruppate non vengono restituite altrove nella risposta. Tipo: stringa Impostazione predefinita: nessuna	No

Risposta di successo

Questa risposta presuppone che il bucket contenga le seguenti chiavi:

```
notes/todos.txt
notes/2005-05-23/customer_mtg_notes.txt
notes/2005-05-23/phone_notes.txt
notes/2005-05-28/sales_notes.txt
```

Sintassi

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>backups</Name>
  <Prefix>notes/</Prefix>
  <MaxKeys>1000</MaxKeys>
  <Delimiter>/</Delimiter>
  <IsTruncated>>false</IsTruncated>
  <Contents>
    <Key>notes/todos.txt</Key>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fd96f00ad9f27c383fc9ac7f"</ETag>
    <Size>5126</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
      <DisplayName>webfile</DisplayName>
```

```
</Owner>
<StorageClass>STANDARD</StorageClass>
</Contents>
<CommonPrefixes>
  <Prefix>notes/2005-05-23/</Prefix>
</CommonPrefixes>
<CommonPrefixes>
  <Prefix>notes/2005-05-28/</Prefix>
</CommonPrefixes>
</ListBucketResult>
```

Come puoi vedere, molti campi della risposta rispecchiano i parametri della richiesta. `IsTruncated` e `CommonPrefixes` sono gli unici elementi di risposta che possono contenere nuove informazioni.

`Contents`

Elementi di risposta

Nome	Description
<code>Contents</code>	Metadati relativi a ciascun oggetto restituito. Tipo: metadati XML Antenato: <code>ListBucketResult</code>
<code>CommonPrefixes</code>	Una risposta può contenere <code>CommonPrefixes</code> solo se si specifica <code>delimiter</code> . Quando lo fai, <code>CommonPrefixes</code> contiene tutte le chiavi (se presenti) comprese tra <code>Prefix</code> e l'occorrenza successiva della stringa specificata da <code>delimiter</code> . In effetti, <code>CommonPrefixes</code> elenca le chiavi che agiscono come sottodirectory nella directory specificata da <code>Prefix</code> . Ad esempio, se <code>prefix</code> è <code>notes/</code> ed <code>delimiter</code> è una barra (/), in <code>notes/summer/july</code> , il prefisso comune è <code>notes/summer/</code> . Tipo: stringa Antenato: <code>ListBucketResult</code>
<code>Delimiter</code>	Fa sì che le chiavi che contengono la stessa stringa tra il prefisso e la prima occorrenza del delimitatore vengano raggruppate in un unico elemento risultato della raccolta. <code>CommonPrefixes</code> Queste chiavi raggruppate non vengono restituite altrove nella risposta.

Nome	Description
	Tipo: stringa Antenato: ListBucketResult
IsTruncated	Specifica se (vero) o meno (falso) sono stati restituiti tutti i risultati. Non è possibile restituire tutti i risultati se il numero di risultati supera quello specificato da MaxKeys Tipo: stringa Antenato: booleano
Marker	Indica da dove iniziare l'inserzione nel bucket. Tipo: stringa Antenato: ListBucketResult
MaxKeys	Il numero massimo di chiavi restituite nel corpo della risposta. Tipo: stringa Antenato: ListBucketResult
Name	Nome del secchio. Tipo: stringa Antenato: ListBucketResult
Prefix	Tasti che iniziano con il prefisso indicato. Tipo: stringa Antenato: ListBucketResult

Corpo della risposta

Per informazioni sulla risposta all'elenco, vedere [Listing Keys Response](#).

Controllo degli accessi

Per elencare le chiavi di un bucket è necessario che sia stato concesso READ l'accesso al bucket.

GetBucketAccessControlPolicy (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

L'GetBucketAccessControlPolicy operazione recupera la politica di controllo degli accessi per un bucket.

Example

Questo esempio recupera la politica di controllo degli accessi per il bucket «quotes».

Sample Request

```
<GetBucketAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetBucketAccessControlPolicy>
```

Sample Response

```
<AccessControlPolicy>
  <Owner>
    <ID>a9a7b886d6fd2441bf9b1c61be666e9</ID>
    <DisplayName>chriscustomer</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6fd2441bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

```
</Grant>
<Grant>
  <Grantee xsi:type="Group">
    <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
  </Grantee>
  <Permission>READ</Permission>
</Grant>
</AccessControlList>
<AccessControlPolicy>
```

Corpo della risposta

La risposta contiene la politica di controllo degli accessi per il bucket. Per una spiegazione di questa risposta, vedere [SOAP Access Policy](#).

Controllo degli accessi

È necessario disporre READ_ACP dei diritti sul bucket per recuperare la politica di controllo degli accessi per un bucket.

SetBucketAccessControlPolicy (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

L'operazione `SetBucketAccessControlPolicy` imposta la politica di controllo degli accessi per un bucket esistente. In caso di successo, la precedente politica di controllo degli accessi per il bucket viene completamente sostituita dalla politica di controllo degli accessi specificata.

Example

Concedi all'utente specificato (di solito il proprietario) FULL_CONTROL l'accesso al bucket «quotes».

Sample Request

```
<SetBucketAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AccessControlList>
```

```
<Grant>
  <Grantee xsi:type="CanonicalUser">
    <ID>a9a7b8863000e241bf9b1c61be666e9</ID>
    <DisplayName>chriscustomer</DisplayName>
  </Grantee>
  <Permission>FULL_CONTROL</Permission>
</Grant>
</AccessControlList>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</SetBucketAccessControlPolicy >
```

Sample Response

```
<GetBucketAccessControlPolicyResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <GetBucketAccessControlPolicyResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </GetBucketAccessControlPolicyResponse>
</GetBucketAccessControlPolicyResponse>
```

Controllo degli accessi

È necessario disporre WRITE_ACP dei diritti sul bucket per impostare la politica di controllo degli accessi per un bucket.

GetBucketLoggingStatus (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

GetBucketLoggingStatusRecupera lo stato di registrazione per un bucket esistente.

Per un'introduzione generale a questa funzionalità, vedere [Server Logs](#).

Example

Sample Request

```
<?xml version="1.0" encoding="utf-8"?>
  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
    <soap:Body>
      <GetBucketLoggingStatus xmlns="https://doc.s3.amazonaws.com/2006-03-01">
        <Bucket>mybucket</Bucket>
        <AWSAccessKeyId>YOUR_AWS_ACCESS_KEY_ID</AWSAccessKeyId>
        <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
        <Signature>YOUR_SIGNATURE_HERE</Signature>
      </GetBucketLoggingStatus>
    </soap:Body>
  </soap:Envelope>
```

Sample Response

```
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" >
    <soapenv:Header>
    </soapenv:Header>
    <soapenv:Body>
      <GetBucketLoggingStatusResponse xmlns="https://s3.amazonaws.com/
doc/2006-03-01">
        <GetBucketLoggingStatusResponse>
          <LoggingEnabled>
            <TargetBucket>mylogs</TargetBucket>
            <TargetPrefix>mybucket-access_log-</TargetPrefix>
          </LoggingEnabled>
        </GetBucketLoggingStatusResponse>
      </GetBucketLoggingStatusResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

Controllo degli accessi

Solo il proprietario di un bucket può richiamare questa operazione.

SetBucketLoggingStatus (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

L'SetBucketLoggingStatus operazione aggiorna lo stato di registrazione di un bucket esistente.

Per un'introduzione generale a questa funzionalità, vedere [Server Logs](#).

Example

Questa richiesta di esempio abilita la registrazione degli accessi al server per il bucket 'mybucket' e configura i log da inviare a 'mylogs' con il prefisso 'access_log-'

Sample Request

```
<?xml version="1.0" encoding="utf-8"?>
  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
    <soap:Body>
      <SetBucketLoggingStatus xmlns="https://doc.s3.amazonaws.com/2006-03-01">
        <Bucket>myBucket</Bucket>
        <AWSAccessKeyId>YOUR_AWS_ACCESS_KEY_ID</AWSAccessKeyId>
        <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
        <Signature>YOUR_SIGNATURE_HERE</Signature>
        <BucketLoggingStatus>
          <LoggingEnabled>
            <TargetBucket>mylogs</TargetBucket>
            <TargetPrefix>mybucket-access_log-</TargetPrefix>
          </LoggingEnabled>
        </BucketLoggingStatus>
      </SetBucketLoggingStatus>
    </soap:Body>
  </soap:Envelope>
```

Sample Response

```
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" >
    <soapenv:Header>
    </soapenv:Header>
    <soapenv:Body>
      <SetBucketLoggingStatusResponse xmlns="https://s3.amazonaws.com/
doc/2006-03-01"/>
    </soapenv:Body>
  </soapenv:Envelope>
```

Controllo degli accessi

Solo il proprietario di un bucket può richiamare questa operazione.

Operazioni sugli oggetti (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

Questa sezione descrive le operazioni che è possibile eseguire sugli oggetti Amazon S3.

Argomenti

- [PutObjectInline \(API SOAP\)](#)
- [PutObject \(API SOAP\)](#)
- [CopyObject \(API SOAP\)](#)
- [GetObject \(API SOAP\)](#)
- [GetObjectExtended \(API SOAP\)](#)
- [DeleteObject \(API SOAP\)](#)
- [GetObjectAccessControlPolicy \(API SOAP\)](#)
- [SetObjectAccessControlPolicy \(API SOAP\)](#)

PutObjectInline (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

L'PutObjectInline operazione aggiunge un oggetto a un bucket. I dati per l'oggetto sono forniti nel corpo del messaggio SOAP.

Se un oggetto esiste già in un bucket, il nuovo oggetto lo sovrascriverà perché Amazon S3 memorizza l'ultima richiesta di scrittura. Tuttavia, Amazon S3 è un sistema distribuito. Se Amazon S3 riceve più richieste di scrittura per lo stesso oggetto quasi contemporaneamente, tutti gli oggetti potrebbero essere archiviati, anche se alla fine solo uno vince. Amazon S3 non prevede il blocco degli oggetti; se ne hai bisogno, assicurati di integrarlo nel livello dell'applicazione.

Per assicurarti che un oggetto non sia danneggiato sulla rete, puoi calcolare l'MD5 di un oggetto, INSERIRLO in Amazon S3 e confrontare l'Etag restituito con il valore MD5 calcolato.

PutObjectInline non è adatto all'uso con oggetti di grandi dimensioni. Il sistema limita questa operazione a lavorare con oggetti di dimensioni pari o inferiori a 1 MB. PutObjectInline avrà esito negativo con il codice di `InlineDataTooLargeError` stato se il parametro `Data` codifica un oggetto di dimensioni superiori a 1 MB. Per caricare oggetti di grandi dimensioni, valuta la possibilità di utilizzare invece l'API non in linea o l' `PutObject` API REST.

Example

Questo esempio scrive del testo e dei metadati nell'oggetto «Nelson» nel bucket «quotes», fornisce a un utente (di solito il proprietario) l'`FULL_CONTROL` accesso all'oggetto e lo rende leggibile da parti anonime.

Sample Request

```
<PutObjectInline xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
```



```

    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>Muntz</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6fde241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>

```

Sample Response

```

<PutObjectInlineResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
    <LastModified>2006-01-01T12:00:00.000Z</lastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>

```

Elementi

- **Bucket**: Il bucket in cui aggiungere l'oggetto.
- **Key**: La chiave da assegnare all'oggetto.

- **Metadata**: È possibile fornire coppie di metadati nome-valore nell'elemento metadati. Questi verranno archiviati con l'oggetto.
- **Data**: La forma codificata in base 64 dei dati.
- **ContentLength**: La lunghezza dei dati in byte.
- **AccessControlList**: Un elenco di controllo degli accessi per la risorsa. Questo elemento è facoltativo. Se omissso, al richiedente viene concesso FULL_CONTROL l'accesso all'oggetto. Se l'oggetto esiste già, la politica di controllo degli accessi preesistente viene sostituita.

Risposte

- **ETag**: Il tag di entità è un hash MD5 dell'oggetto che è possibile utilizzare per eseguire recuperi condizionali dell'oggetto utilizzando `GetObjectExtended`. L'ETag riflette solo le modifiche al contenuto di un oggetto, non i suoi metadati.
- **LastModified**: Il timestamp di Amazon S3 per l'oggetto salvato.

Controllo degli accessi

È necessario avere WRITE accesso al bucket per inserire oggetti nel bucket.

Risorse correlate

- [PutObject \(API SOAP\)](#)
- [CopyObject \(API SOAP\)](#)

PutObject (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

L'PutObject operazione aggiunge un oggetto a un bucket. I dati dell'oggetto sono allegati come allegato DIME.

Per assicurarti che un oggetto non sia danneggiato sulla rete, puoi calcolare l'MD5 di un oggetto, INSERIRLO in Amazon S3 e confrontare l'Etag restituito con il valore MD5 calcolato.

Se un oggetto esiste già in un bucket, il nuovo oggetto lo sovrascriverà perché Amazon S3 memorizza l'ultima richiesta di scrittura. Tuttavia, Amazon S3 è un sistema distribuito. Se Amazon S3 riceve più richieste di scrittura per lo stesso oggetto quasi contemporaneamente, tutti gli oggetti potrebbero essere archiviati, anche se alla fine solo uno vince. Amazon S3 non prevede il blocco degli oggetti; se ne hai bisogno, assicurati di integrarlo nel livello dell'applicazione.

Example

Questo esempio inserisce alcuni dati e metadati nell'oggetto «Nelson» del bucket «quotes», consente a un utente (di solito il proprietario) di FULL_CONTROL accedere all'oggetto e lo rende leggibile da parti anonime. In questo esempio, l'allegato effettivo non viene mostrato.

Sample Request

```
<PutObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>Muntz</Value>
  </Metadata>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers<URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</PutObject>
```

```

    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2007-05-11T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObject>

```

Sample Response

```

<PutObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2006-03-01T12:00:00.183Z</LastModified>
  </PutObjectResponse>
</PutObjectResponse>

```

Elementi

- **Bucket**: Il bucket in cui aggiungere l'oggetto.
- **Key**: La chiave da assegnare all'oggetto.
- **Metadata**: È possibile fornire coppie di metadati nome-valore nell'elemento metadati. Questi verranno archiviati con l'oggetto.
- **ContentLength**: La lunghezza dei dati in byte.
- **AccessControlList**: Un elenco di controllo degli accessi per la risorsa. Questo elemento è facoltativo. Se omissso, al richiedente viene concesso FULL_CONTROL l'accesso all'oggetto. Se l'oggetto esiste già, la politica di controllo degli accessi preesistente viene sostituita.

Risposte

- **ETag**: Il tag di entità è un hash MD5 dell'oggetto che è possibile utilizzare per eseguire recuperi condizionali dell'oggetto utilizzando. GetObjectExtended L'ETag riflette solo le modifiche al contenuto di un oggetto, non i suoi metadati.
- **LastModified**: Il timestamp di Amazon S3 per l'oggetto salvato.

Controllo degli accessi

Per inserire oggetti in un bucket, devi avere WRITE accesso al bucket.

Risorse correlate

- [CopyObject \(API SOAP\)](#)

CopyObject (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

Description

L'CopyObject operazione crea una copia di un oggetto quando si specificano la chiave e il bucket di un oggetto sorgente e la chiave e il bucket di una destinazione di destinazione.

Quando si copia un oggetto, è possibile conservare tutti i metadati (impostazione predefinita) o specificarne di nuovi. Tuttavia, l'ACL non viene conservato ed è impostato `private` per l'utente che effettua la richiesta. Per sovrascrivere l'impostazione ACL predefinita, specifica un nuovo ACL quando generi una richiesta di copia. Per ulteriori informazioni, consulta [Gestione degli accessi con le ACL](#).

Tutte le richieste di copia devono essere autenticate. Inoltre, è necessario disporre dell'accesso in lettura all'oggetto sorgente e dell'accesso in scrittura al bucket di destinazione. Per ulteriori informazioni, vedere [Utilizzo dell'accesso di autenticazione](#).

Per copiare un oggetto solo in determinate condizioni, ad esempio se l'Etag corrisponde o se l'oggetto è stato modificato prima o dopo una data specificata, usa i parametri di richiesta `CopySourceIfUnmodifiedSince`, `CopyIfUnmodifiedSinceCopySourceIfMatch`, o `CopySourceIfNoneMatch`.

Note

Potrebbe essere necessario configurare il timeout del socket dello stack SOAP per copiare oggetti di grandi dimensioni.

Sintassi della richiesta

```
<CopyObject xmlns="http://bucket_name.s3.amazonaws.com/2006-03-01">
  <SourceBucket>source_bucket</SourceBucket>
  <SourceObject>source_object</SourceObject>
  <DestinationBucket>destination_bucket</DestinationBucket>
  <DestinationObject>destination_object</DestinationObject>
  <MetadataDirective>{REPLACE | COPY}</MetadataDirective>
  <Metadata>
    <Name>metadata_name</Name>
    <Value>metadata_value</Value>
  </Metadata>
  ...
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="user_type">
        <ID>user_id</ID>
        <DisplayName>display_name</DisplayName>
      </Grantee>
      <Permission>permission</Permission>
    </Grant>
    ...
  </AccessControlList>
  <CopySourceIfMatch>etag</CopySourceIfMatch>
  <CopySourceIfNoneMatch>etag</CopySourceIfNoneMatch>
  <CopySourceIfModifiedSince>date_time</CopySourceIfModifiedSince>
  <CopySourceIfUnmodifiedSince>date_time</CopySourceIfUnmodifiedSince>
  <AWSAccessKeyId>AWSAccessKeyId</AWSAccessKeyId>
  <Timestamp>TimeStamp</Timestamp>
  <Signature>Signature</Signature>
</CopyObject>
```

Parametri della richiesta

Nome	Description	Richiesto
SourceBucket	Il nome del bucket di origine. Tipo: stringa Impostazione predefinita: nessuna	Sì

Nome	Description	Richiesto
	Vincoli: un bucket di origine valido.	
SourceKey	Il nome chiave dell'oggetto sorgente. Tipo: stringa Impostazione predefinita: nessuna Vincoli: la chiave per un oggetto sorgente valido a cui si dispone dell'accesso in LETTURA.	Sì
DestinationBucket	Nome del bucket di destinazione. Tipo: stringa Impostazione predefinita: nessuna Vincoli: è necessario disporre dell'accesso WRITE al bucket di destinazione.	Sì
DestinationKey	La chiave dell'oggetto di destinazione. Tipo: stringa Impostazione predefinita: nessuna Vincoli: è necessario disporre dell'accesso WRITE al bucket di destinazione.	Sì

Nome	Description	Richiesto
MetadataDirective	Specifica se i metadati vengono copiati dall'oggetto di origine o sostituiti con i metadati forniti nella richiesta. Tipo: stringa Predefinito: COPY Valori validi: COPY REPLACE Vincoli: valori diversi COPY o REPLACE genereranno un errore immediato. Non è possibile copiare un oggetto su se stesso a meno che l' MetadataDirective intestazione non sia specificata e il relativo valore impostato su. REPLACE	No
Metadata	Specifica le coppie nome-valore dei metadati da impostare per l'oggetto. Se MetadataDirective è impostato su, tutti i metadati vengono ignorati. COPY Tipo: stringa Impostazione predefinita: nessuna Vincoli: nessuno.	No
AccessControlList	Consente l'accesso agli utenti tramite indirizzi e-mail o ID utente canonico. Tipo: stringa Impostazione predefinita: nessuna Vincoli: nessuno	No

Nome	Description	Richiesto
CopySourceIfMatch	Copia l'oggetto se il relativo tag di entità (ETag) corrisponde al tag specificato; altrimenti restituisce un. PreconditionFailed Tipo: stringa Impostazione predefinita: nessuna Vincoli: nessuno. Se l'Etag non corrisponde, l'oggetto non viene copiato.	No
CopySourceIfNoneMatch	Copia l'oggetto se il relativo tag di entità (ETag) è diverso dall'Etag specificato; altrimenti restituisce un errore. Tipo: stringa Impostazione predefinita: nessuna Vincoli: nessuno.	No
CopySourceIfUnmodifiedSince	Copia l'oggetto se non è stato modificato dall'ora specificata; altrimenti restituisce un PreconditionFailed. Tipo: DateTime Impostazione predefinita: nessuna	No
CopySourceIfModifiedSince	Copia l'oggetto se è stato modificato dall'ora specificata; altrimenti restituisce un errore. Tipo: DateTime Impostazione predefinita: nessuna	No

Sintassi della risposta

```
<CopyObjectResponse xmlns="http://bucket_name.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"etag"</ETag>
    <LastModified>timestamp</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

Elementi di risposta

Di seguito è riportato un elenco di elementi di risposta.

Note

L'API SOAP non restituisce spazi bianchi aggiuntivi. Gli spazi bianchi aggiuntivi vengono restituiti solo dall'API REST.

Nome	Description
Etag	Restituisce l'etag del nuovo oggetto. L'ETag riflette solo le modifiche al contenuto di un oggetto, non i suoi metadati. Tipo: stringa Antenato: CopyObjectResult
LastModified	Restituisce la data dell'ultima modifica dell'oggetto. Tipo: stringa Antenato: CopyObjectResult

Per informazioni sugli elementi di risposta generali, vedere [Utilizzo delle intestazioni](#) di risposta agli errori REST.

Errori speciali

Non ci sono errori speciali per questa operazione. Per informazioni sugli errori generali di Amazon S3, consulta [Elenco dei codici di errore](#).

Esempi

In questo esempio viene copiato l'oggetto `flotsam` dal bucket `pacific` all'oggetto `jetsam` del bucket `atlantic` conservandone i metadata.

Richiesta di esempio

```
<CopyObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <SourceBucket>pacific</SourceBucket>
  <SourceObject>flotsam</SourceObject>
  <DestinationBucket>atlantic</DestinationBucket>
  <DestinationObject>jetsam</DestinationObject>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2008-02-18T13:54:10.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbq7RrtSFmw=</Signature>
</CopyObject>
```

Risposta di esempio

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fd9a96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

Questo esempio copia l'oggetto «tweedledee» dal bucket `Wonderland` all'oggetto «tweedledum» del bucket `Wonderland`, sostituendone i metadata.

Richiesta di esempio

```
<CopyObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <SourceBucket>wonderland</SourceBucket>
  <SourceObject>tweedledee</SourceObject>
  <DestinationBucket>wonderland</DestinationBucket>
  <DestinationObject>tweedledum</DestinationObject>
  <MetadataDirective>REPLACE</MetadataDirective>
</CopyObject>
```

```
<Metadata>
  <Name>Content-Type</Name>
  <Value>text/plain</Value>
</Metadata>
<Metadata>
  <Name>relationship</Name>
  <Value>twins</Value>
</Metadata>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2008-02-18T13:54:10.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbq7RrtSFmw=</Signature>
</CopyObject>
```

Risposta di esempio

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

Risorse correlate

- [PutObject \(API SOAP\)](#)
- [PutObjectInline \(API SOAP\)](#)

GetObject (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

L'GetObject operazione restituisce la versione corrente di un oggetto. Se si tenta di utilizzare GetObject un oggetto con un indicatore di eliminazione come versione corrente, S3 restituisce un errore 404. Non è possibile utilizzare l'API SOAP per recuperare una versione specificata di un oggetto. Per farlo, usa l'API REST. Per ulteriori informazioni, vedere [Controlla le versioni](#). Per ulteriori opzioni, utilizzare l'[GetObjectExtended \(API SOAP\)](#) operazione.

 Note

I nomi delle chiavi dell'oggetto con il valore "soap" non sono supportati per [richieste in stile hosting virtuale](#). Per i valori dei nomi delle chiavi dell'oggetto in cui viene utilizzato "soap", deve invece essere usato un [URL in stile percorso](#).

Example

Questo esempio ottiene l'oggetto «Nelson» dal bucket «quotes».

Sample Request

```
<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Sample Response

```
<GetObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>Muntz</Value>
    </Metadata>
    <Data>aGEtaGE=</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
  </GetObjectResponse>
</GetObjectResponse>
```

```
<ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"></ETag>
</GetObjectResponse>
</GetObjectResponse>
```

Elementi

- **Bucket** : Il bucket da cui recuperare l'oggetto.
- **Key** : La chiave che identifica l'oggetto.
- **GetMetadata** : I metadati vengono restituiti con l'oggetto se ciò è vero.
- **GetData** : I dati dell'oggetto vengono restituiti se questo è vero.
- **InlineData** : Se questo è vero, i dati vengono restituiti, con codifica in base 64, come parte del corpo SOAP della risposta. Se false, i dati vengono restituiti come allegato SOAP. L' **InlineData** opzione non è adatta all'uso con oggetti di grandi dimensioni. Il sistema limita questa operazione a lavorare con 1 MB di dati o meno. Una **GetObject** richiesta con il **InlineData** flag impostato avrà esito negativo con il codice di **InlineDataTooLargeError** stato se il parametro **Data** risultante è stato codificato per più di 1 MB. Per scaricare oggetti di grandi dimensioni, valuta la possibilità di chiamare **GetObject** senza impostare il **InlineData** flag o di utilizzare invece l'API REST.

Elementi restituiti

- **Metadata** : I metadati associati nome-valore memorizzati con l'oggetto.
- **Data** : Se **InlineData** era vero nella richiesta, contiene i dati dell'oggetto codificati in base 64.
- **LastModified** : L'ora in cui l'oggetto è stato archiviato in Amazon S3.
- **ETag** : Il tag di entità dell'oggetto. Si tratta di un hash dell'oggetto che può essere utilizzato per eseguire risultati condizionali. L'**ETag** riflette solo le modifiche al contenuto di un oggetto, non i suoi metadati.

Controllo degli accessi

Puoi leggere un oggetto solo se ti è stato concesso **READ** l'accesso all'oggetto.

Download SOAP suddivisi in blocchi e ripristinabili

Per offrire **GET** flessibilità, Amazon S3 supporta download in blocchi e ripristinabili.

Seleziona tra le seguenti opzioni:

- Per i download di oggetti di grandi dimensioni, potresti volerli suddividere in blocchi più piccoli. Per ulteriori informazioni, consulta [Intervallo GET](#)
- Per GET le operazioni che falliscono, puoi progettare l'applicazione in modo che scarichi il resto anziché l'intero file. Per ulteriori informazioni, consulta [Ripristino degli errori REST GET](#)

Intervallo GET

Per alcuni clienti, potresti voler suddividere i download di grandi dimensioni in download più piccoli. Per suddividere un GET in unità più piccole, usa Range.

Prima di poter suddividere un GET in unità più piccole, è necessario determinarne le dimensioni. Ad esempio, la richiesta seguente ottiene la dimensione dell'oggetto bigfile.

```
<ListBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Prefix>bigfile</Prefix>
  <MaxKeys>1</MaxKeys>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListBucket>
```

Amazon S3 restituisce la seguente risposta.

```
<ListBucketResult xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <Name>quotes</Name>
  <Prefix>N</Prefix>
  <MaxKeys>1</MaxKeys>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>bigfile</Key>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <Size>2023276</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>bcaf1ffd86f41161ca5fb16fd081034f</ID>
      <DisplayName>bigfile</DisplayName>
    </Owner>
  </Contents>
</ListBucketResult>
```

Segue una richiesta che scarica il primo megabyte dall'oggetto bigfile.

```
<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Key>bigfile</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <ByteRangeStart>0</ByteRangeStart>
  <ByteRangeEnd>1048576</ByteRangeEnd>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Amazon S3 restituisce il primo megabyte del file e l'Etag del file.

```
<GetObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>Muntz</Value>
    </Metadata>
    <Data>--first megabyte of bigfile--</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>
```

Per assicurarti che il file non sia cambiato dopo il download della parte precedente, specifica l'elemento. IfMatch Sebbene l' IfMatch elemento non sia obbligatorio, è consigliato per contenuti suscettibili di modifiche.

Quella che segue è una richiesta che ottiene il resto del file, utilizzando l'intestazione della `IfMatch` richiesta.

```
<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Key>bigfile</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <ByteRangeStart>10485761</ByteRangeStart>
  <ByteRangeEnd>2023276</ByteRangeEnd>
  <IfMatch>"828ef3fdfa96f00ad9f27c383fc9ac7f"</IfMatch>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Amazon S3 restituisce la seguente risposta e il resto del file.

```
<GetObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>>Muntz</Value>
    </Metadata>
    <Data>--remainder of bigfile--</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>
```

Con versione `GetObject`

La seguente richiesta restituisce la versione specificata dell'oggetto nel bucket.

```
<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Risposta di esempio

```
<GetObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>Muntz</Value>
    </Metadata>
    <Data>aGEtaGE=</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>
```

Ripristino degli errori REST GET

Se un oggetto GET fallisce, puoi recuperare il resto del file specificando l'intervallo da scaricare. A tale scopo, è necessario ottenere la dimensione dell'oggetto utilizzando `ListBucket` ed eseguire un intervallo GET sul resto del file. Per ulteriori informazioni, consulta [GetObjectExtended \(API SOAP\)](#).

Risorse correlate

[Operazioni sugli oggetti \(API SOAP\)](#)

GetObjectExtended (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

GetObjectExtended è esattamente lo stesso [GetObject \(API SOAP\)](#), tranne per il fatto che supporta i seguenti elementi aggiuntivi che possono essere utilizzati per ottenere gran parte delle stesse funzionalità fornite dalle intestazioni HTTP GET (vai a). <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>

GetObjectExtended supporta i seguenti elementi oltre a quelli supportati da: GetObject

- **ByteRangeStart, ByteRangeEnd**: Questi elementi specificano che deve essere recuperata solo una parte dei dati dell'oggetto. Seguono il comportamento degli intervalli di byte HTTP (vai a [http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html #sec14.35](http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35)).
- **IfModifiedSince**: Restituisce l'oggetto solo se il timestamp dell'oggetto è successivo al timestamp specificato. (#sec14.25) <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>
- **IfUnmodifiedSince**: Restituisce l'oggetto solo se il timestamp dell'oggetto è precedente o uguale al timestamp specificato. (vai a #sec14.28) <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>
- **IfMatch**: Restituisce l'oggetto solo se il relativo ETag corrisponde ai tag forniti. (vai a [http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html #sec14.24](http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.24))
- **IfNoneMatch**: Restituisce l'oggetto solo se il relativo ETag non corrisponde ai tag forniti. (vai a [http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html #sec14.26](http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.26))
- **ReturnCompleteObjectOnConditionFailure**: ReturnCompleteObjectOnConditionFailure: se è vera, se la richiesta include un elemento di intervallo e uno o entrambi IfUnmodifiedSince/IfMatch gli elementi e la condizione fallisce, restituisce l'intero oggetto anziché un errore. Ciò abilita la If-Range funzionalità (vai a [http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html #sec14.27](http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.27)).

DeleteObject (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

L'DeleteObject operazione rimuove l'oggetto specificato da Amazon S3. Una volta eliminato, non esiste alcun metodo per ripristinare o ripristinare l'eliminazione di un oggetto.

Note

Se elimini un oggetto che non esiste, Amazon S3 restituirà un risultato positivo (non un messaggio di errore).

Example

Questo esempio elimina l'oggetto «Nelson» dal bucket «quotes».

Sample Request

```
<DeleteObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AWSAccessKeyId> AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</DeleteObject>
```

Sample Response

```
<DeleteObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <DeleteObjectResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </DeleteObjectResponse>
</DeleteObjectResponse>
```

Elementi

- **Bucket** : Il secchio che contiene l'oggetto.
- **Key** : La chiave che identifica l'oggetto.

Controllo degli accessi

Puoi eliminare un oggetto solo se hai **WRITE** accesso al bucket, indipendentemente dal proprietario dell'oggetto o dai diritti che gli vengono concessi.

GetObjectAccessControlPolicy (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

L'GetObjectAccessControlPolicy operazione recupera la politica di controllo degli accessi per un oggetto.

Example

Questo esempio recupera la politica di controllo degli accessi per l'oggetto «Nelson» dal bucket «quotes».

Sample Request

```
<GetObjectAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObjectAccessControlPolicy>
```

Sample Response

```
<AccessControlPolicy>
  <Owner>
```

```
<ID>a9a7b886d6fd24a541bf9b1c61be666e9</ID>
<DisplayName>chriscustomer</DisplayName>
</Owner>
<AccessControlList>
  <Grant>
    <Grantee xsi:type="CanonicalUser">
      <ID>a9a7b841bf9b1c61be666e9</ID>
      <DisplayName>chriscustomer</DisplayName>
    </Grantee>
    <Permission>FULL_CONTROL</Permission>
  </Grant>
  <Grant>
    <Grantee xsi:type="Group">
      <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
    </Grantee>
    <Permission>READ</Permission>
  </Grant>
</AccessControlList>
</AccessControlPolicy>
```

Corpo della risposta

La risposta contiene la politica di controllo degli accessi per il bucket. Per una spiegazione di questa risposta, [SOAP Access Policy](#).

Controllo degli accessi

È necessario disporre READ_ACP dei diritti sull'oggetto per recuperare la politica di controllo dell'accesso per un oggetto.

SetObjectAccessControlPolicy (API SOAP)

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

L'operazione `SetObjectAccessControlPolicy` imposta la politica di controllo dell'accesso per un oggetto esistente. In caso di successo, la precedente politica di controllo degli accessi per l'oggetto viene completamente sostituita dalla politica di controllo degli accessi specificata.

Example

Questo esempio consente all'utente specificato (in genere al proprietario) di FULL_CONTROL accedere all'oggetto «Nelson» dal bucket «quotes».

Sample Request

```
<SetObjectAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</SetObjectAccessControlPolicy>
```

Sample Response

```
<SetObjectAccessControlPolicyResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <SetObjectAccessControlPolicyResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </SetObjectAccessControlPolicyResponse>
</SetObjectAccessControlPolicyResponse>
```

Chiave

Controllo degli accessi

È necessario disporre WRITE_ACP dei diritti sull'oggetto per impostare la politica di controllo degli accessi per un bucket.

Autenticazione delle richieste SOAP

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

Per attestare l'identità dell'entità che effettua la richiesta, ogni richiesta non anonima deve contenere informazioni di autenticazione. In SOAP le informazioni di autenticazione sono inserite nei seguenti elementi della richiesta SOAP:

- AWS L'ID della tua chiave di accesso

Note

Quando si effettuano richieste SOAP autenticate; le credenziali di sicurezza temporanee non sono supportate. Per ulteriori informazioni sui tipi di credenziali, consultare [Esecuzione di richieste](#).

- **Timestamp:** Deve essere un DateTime (vai a <http://www.w3.org/TR/xmlschema-2/#dateTime>) nel fuso orario Coordinated Universal Time (Greenwich Mean Time), ad esempio. 2009-01-01T12:00:00.000Z L'autorizzazione non riuscirà se la differenza tra il timestamp e l'orologio nei server Amazon S3 è maggiore di 15 minuti.
- **Signature:** Il HMAC-SHA1 riassunto RFC 2104 (vai a <http://www.ietf.org/rfc/rfc2104.txt>) della concatenazione di «AmazonS3" + OPERATION + Timestamp, utilizzando la tua chiave di accesso segreta come chiave. AWS Ad esempio, nella seguente richiesta di esempio, l'elemento di firma CreateBucket conterrebbe il riassunto del valore "«: HMAC-SHA1 AmazonS3CreateBucket2009-01-01T12:00:00.000Z

Ad esempio, nella seguente richiesta di CreateBucket esempio, l'elemento signature conterrebbe il HMAC-SHA1 riassunto del valore "«AmazonS3CreateBucket2009-01-01T12:00:00.000Z:

Example

```
<CreateBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Acl>private</Acl>
```



```
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2009-01-01T12:00:00.000Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</CreateBucket>
```

Note

Le richieste SOAP, autenticate e anonime, devono essere inviate ad Amazon S3 tramite SSL. Amazon S3 restituisce un errore quando invii una richiesta SOAP tramite HTTP.

Important

A causa delle diverse interpretazioni riguardo a come ridurre la precisione del tempo aggiuntivo, gli utenti .NET devono evitare di inviare ad Amazon S3 timestamp troppo specifici. Ciò è possibile strutturando manualmente gli oggetti DateTime esclusivamente con una precisione pari al millisecondo.

Impostazione della policy d'accesso con SOAP

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

Il controllo dell'accesso può essere impostato al momento della scrittura di un bucket o di un oggetto includendo l'elemento AccessControlList "con la richiesta a CreateBucketPutObjectInline, o. PutObject L' AccessControlList elemento è descritto in [Identity and Access Management for Amazon S3](#). Se non viene specificata alcuna lista di controllo degli accessi con queste operazioni, la risorsa viene creata con una politica di accesso predefinita che consente al richiedente l'accesso FULL_CONTROL (questo vale anche se la richiesta è una PutObject richiesta PutObjectInline o per un oggetto già esistente).

Di seguito viene riportata una richiesta che scrive dati in un oggetto, lo rende leggibile da parte di entità principali anonime e concede all'utente specificato i diritti FULL_CONTROL sul bucket

(la maggior parte degli sviluppatori si assegna l'accesso FULL_CONTROL al bucket di cui sono proprietari).

Example

Di seguito è riportata una richiesta che scrive dati in un oggetto e lo rende leggibile da parte di entità principali anonime.

Sample Request

```
<PutObjectInline xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers<URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2009-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>
```

Sample Response

```
<PutObjectInlineResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
```

```
<ETag>"828ef3fd9a96f00ad9f27c383fc9ac7f"</ETag>
<LastModified>2009-01-01T12:00:00.000Z</LastModified>
</PutObjectInlineResponse>
</PutObjectInlineResponse>
```

È possibile leggere o impostare la policy di controllo accessi per un bucket o un oggetto esistente utilizzando i metodi `GetBucketAccessControlPolicy`, `GetObjectAccessControlPolicy`, `SetBucketAccessControlPolicy` e `SetObjectAccessControlPolicy`. Per ulteriori informazioni, consulta la spiegazione dettagliata di questi metodi.

Elementi comuni

In qualsiasi richiesta SOAP è possibile includere i seguenti elementi correlati all'autorizzazione:

- **AWSSignatureVersion**: L'ID della chiave di AWS accesso del richiedente
- **Timestamp**: l'ora corrente del sistema
- **Signature**: la firma per la richiesta

Risposte agli errori SOAP

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli SDK. AWS

In SOAP, il risultato dell'errore viene restituito al client come errore SOAP, con il codice di risposta HTTP 500. Se non si riceve un errore SOAP, la richiesta è stata eseguita correttamente. Il codice di errore SOAP di Amazon S3 è composto da un codice di errore SOAP 1.1 standard («Server» o «Client») concatenato con il codice di errore Amazon. S3-specific Ad esempio: "" o "". Server.InternalError Client.NoSuchBucket L'elemento stringa di errore di SOAP contiene un messaggio di errore generico in inglese leggibile da un interprete umano. Infine, l'elemento del dettaglio di errore SOAP contiene informazioni varie pertinenti all'errore.

Ad esempio, se si tenta di eliminare l'oggetto «Fred», che non esiste, il corpo della risposta SOAP contiene un errore SOAP `NoSuchKey` "".

L'esempio seguente mostra un esempio di risposta di errore SOAP.

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.NoSuchKey</Faultcode>
    <Faultstring>The specified key does not exist.</Faultstring>
    <Detail>
      <Key>Fred</Key>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

La tabella seguente illustra gli elementi di risposta agli errori SOAP

Nome	Description
Detail	Contenitore per la chiave coinvolta nell'errore Tipo: Container Antenato: Body.Fault
Fault	Contenitore per informazioni sugli errori. Tipo: Container Antenato: corpo
Faultcode	Il codice di errore è una stringa che identifica in modo univoco una condizione di errore. Ha lo scopo di essere letta e compresa dai programmi che rilevano e gestiscono gli errori in base al loro tipo. Per ulteriori informazioni, vedere Elenco dei codici di errore. Tipo: stringa Antenato: Body.Fault
Faultstring	La stringa di errore contiene una descrizione generica della condizione di errore in inglese. È destinato a interlocutori umani. I programmi semplici visualizzano il messaggio direttamente all'utente finale se si verifica una condizione di errore che non conoscono o che non sono interessati a gestire. I programmi sofisticati con una gestione degli errori più completa e una corretta internazionalizzazione hanno maggiori probabilità di ignorare la stringa di errore.

Nome	Description
	Tipo: stringa
	Antenato: Body.Fault
Key	Identifica la chiave coinvolta nell'errore
	Tipo: stringa
	Antenato: Body.Fault

Appendice: Autenticazione delle richieste (AWS firma (versione 2))

Important

Questa sezione descrive come autenticare le richieste utilizzando AWS Signature Version 2. Signature Version 2 sta per essere disattivato in quanto obsoleto e Amazon S3 accetterà solo richieste API firmate con Signature Version 4. Per ulteriori informazioni, consulta [AWS Signature Version 2 disattivata \(obsoleta\) per Amazon S3](#)

La versione 4 di Signature è supportata in tutte le Regioni AWS versioni ed è l'unica versione supportata per le nuove regioni. Per ulteriori informazioni, consulta [Authenticating Requests \(AWS Signature Version 4\)](#) nel Amazon Simple Storage Service API Reference.

Amazon S3 offre la possibilità di identificare il protocollo Signature Version dell'API utilizzato per firmare una richiesta. È importante stabilire se qualcuno dei flussi di lavoro sta usando Signature Versione 2 e aggiornarlo in modo che utilizzi Signature Versione 4 per evitare problemi.

- Se stai utilizzando i registri CloudTrail degli eventi (opzione consigliata), consulta [Identificazione delle richieste Amazon S3 Signature versione 2 utilizzando CloudTrail](#) on how to query e identificando tali richieste.
- Se stai utilizzando i log di accesso al server Amazon S3, consulta [Utilizzo dei log di accesso al server Amazon S3 per identificare le richieste](#)

Argomenti

- [Autenticazione delle richieste tramite l'API REST \(AWS firma \(versione 2\)\)](#)

- [Firma e autenticazione delle richieste REST \(AWS firma \(versione 2\)\)](#)
- [Browser-based caricamenti utilizzando POST \(AWS firma \(versione 2\)\)](#)

Autenticazione delle richieste tramite l'API REST (AWS firma (versione 2))

Quando accedi ad Amazon S3 tramite REST, devi fornire gli elementi seguenti nella richiesta, in modo che questa possa essere autenticata:

Elementi della richiesta

- AWS ID della chiave di accesso: ogni richiesta deve contenere l'ID della chiave di accesso dell'identità utilizzata per inviare la richiesta.
- Firma: ogni richiesta deve contenere una firma di richiesta valida o la richiesta viene rifiutata.

Una firma di richiesta viene calcolata utilizzando la chiave di accesso segreta dell'interessato, che è un segreto condiviso unicamente tra quest'ultimo e AWS.

- Timestamp: ogni richiesta deve contenere la data e l'ora in cui è stata creata, sotto forma di stringa in UTC.
- Data: ogni richiesta deve contenere il timestamp.

In funzione dell'operazione API in uso, è possibile fornire un'ora e una data di scadenza per la richiesta al posto del time stamp o in aggiunta a esso. consulta l'argomento sull'autenticazione per l'operazione specifica da eseguire per determinare ciò che è necessario.

Di seguito vengono presentate le fasi generali per l'autenticazione delle richieste in Amazon S3. Si presume che l'interessato disponga degli elementi necessari: credenziali di sicurezza, ID chiave di accesso e Secret Access Key.

You**1 Create a request:****Request**

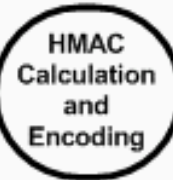
AccessKeyId = ...
Action = ...
Timestamp = ...
ParameterA = ...

2 Create an HMAC-SHA1 signature:

String based on
request contents

+

Your Secret Access Key



Your Signature

3 Send the request and signature to AWS:**Request**

AccessKeyId = ...
Action = ...
Timestamp = ...
ParameterA = ...

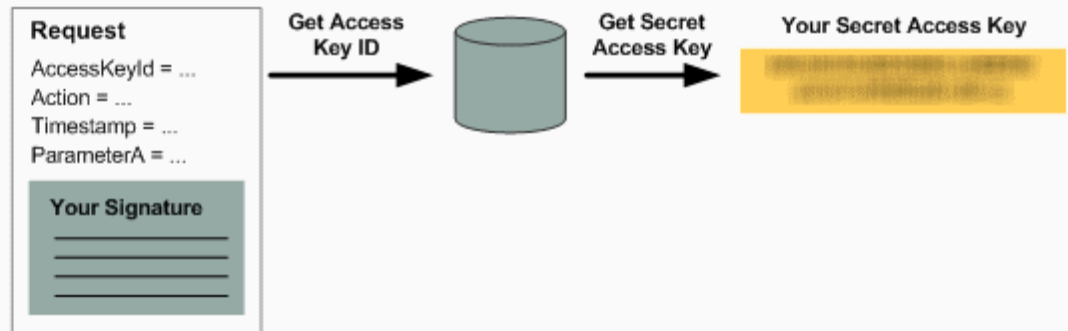
Your Signature

**AWS**

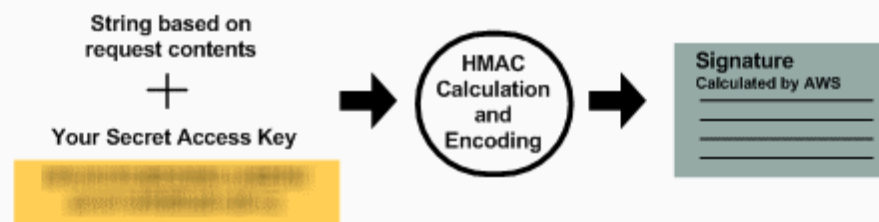
- 1 Crea una richiesta per. AWS
- 2 Calcolare la firma utilizzando la Secret Access Key.
- 3 Inviare la richiesta ad Amazon S3. Includere l'ID chiave di accesso e la firma nella richiesta. Amazon S3 completa le tre fasi successive.

AWS

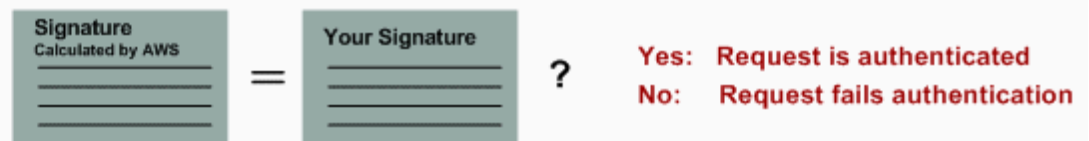
4 Retrieve your Secret Access Key:



5 Create an HMAC-SHA1 signature:



6 Compare the two signatures:



4	Amazon S3 utilizza l'ID chiave di accesso per cercare la chiave di accesso segreta.
5	Amazon S3 calcola una firma dai dati della richiesta e dalla chiave di accesso segreta tramite lo stesso algoritmo che hai utilizzato per calcolare la firma inviata nella richiesta.
6	Se la firma generata da Amazon S3 corrisponde a quella inviata nella richiesta, la richiesta viene considerata autentica. Se il confronto non ha esito positivo, la richiesta viene ignorata e Amazon S3 restituisce una risposta di errore.

Informazioni dettagliate sull'autenticazione

Per le informazioni dettagliate sull'autenticazione REST, consulta [Firma e autenticazione delle richieste REST \(AWS firma \(versione 2\)\)](#).

Firma e autenticazione delle richieste REST (AWS firma (versione 2))

Argomenti

- [Utilizzo di credenziali di sicurezza temporanee](#)
- [Intestazione di autenticazione](#)
- [Standardizzazione della richiesta per la firma](#)
- [Costruzione dell'elemento CanonicalizedResource](#)
- [Costruire l'elemento CanonicalizedAmzHeaders](#)
- [Elementi di intestazione HTTP posizionali e denominati StringToSign](#)
- [Necessità del time stamp](#)
- [Esempi di autenticazione](#)
- [Problemi con la firma delle richieste REST](#)
- [Alternativa per l'autenticazione di una richiesta tramite stringa di query](#)

Note

In questo argomento viene illustrato come autenticare le richieste tramite Signature Version 2. Amazon S3 supporta ora il più recente Signature Version 4. L'ultima versione è ora supportata in tutte le regioni e le regioni create dopo il 30 gennaio 2014 supporteranno solo Signature Version 4. Per ulteriori informazioni, consulta [Authenticating Requests \(AWS Signature Version 4\)](#) nel Amazon Simple Storage Service API Reference.

Per autenticazione si intende il processo di attestazione dell'identità in un sistema. L'identità è un importante fattore per le decisioni di controllo degli accessi in Amazon S3. Le richieste vengono accettate o rifiutate in parte anche in base all'identità del richiedente. Ad esempio, il diritto di creare bucket è riservato agli sviluppatori registrati e, di default, è solo il proprietario del bucket in questione ad avere il diritto di creare oggetti al suo interno. Poiché gli sviluppatori inviano richieste che invocano questi privilegi, dovranno provare la propria identità al sistema autenticando le richieste: in questa sezione viene illustrato come procedere.

 Note

I contenuti di questa sezione non si applicano alle richieste HTTP POST. Per ulteriori informazioni, consulta [Browser-based caricamenti utilizzando POST \(AWS firma \(versione 2\)\)](#).

Per l'autenticazione, l'API REST di Amazon S3 utilizza uno schema HTTP personalizzato basato su un codice HMAC (Hash Message Authentication Code) con chiave. Per autenticare una richiesta, è necessario in primo luogo concatenare alcuni elementi della richiesta per formare una stringa. Quindi utilizzi la tua chiave di accesso AWS segreta per calcolare l'HMAC di quella stringa. Informalmente, questo processo è chiamato «firma della richiesta» e l'output dell'algoritmo HMAC è la firma, poiché simula le proprietà di sicurezza di una firma reale. Infine, la firma viene aggiunta come parametro della richiesta tramite la sintassi descritta in questa sezione.

Quando il sistema riceve una richiesta autenticata, recupera la chiave di accesso AWS segreta che si afferma di avere e la utilizza allo stesso modo per calcolare una firma per il messaggio ricevuto. quindi confronta la firma calcolata con quella presentata dal richiedente. Se le due firme corrispondono, il sistema conclude che il richiedente deve avere accesso alla chiave di accesso AWS segreta e quindi agisce con l'autorità del mandante a cui è stata rilasciata la chiave. Se le due firme non coincidono, la richiesta viene scartata e il sistema risponde con un messaggio di errore.

Example Richiesta REST Amazon S3 autenticata

```
GET /photos/puppy.jpg HTTP/1.1
Host: awsexamplebucket1.us-west-1.s3.amazonaws.com
Date: Tue, 27 Mar 2007 19:36:42 +0000

Authorization: AWS AKIAIOSFODNN7EXAMPLE:
qgk2+6Sv9/oM7G3qLEjTH1a1l1g=
```

Utilizzo di credenziali di sicurezza temporanee

Se la richiesta viene firmata tramite credenziali di sicurezza temporanee (consulta [Esecuzione di richieste](#)), è necessario includere nella richiesta il corrispondente token di sicurezza aggiungendo l'intestazione `x-amz-security-token`.

Quando si ottengono credenziali di sicurezza temporanee utilizzando l'API Servizio di token di sicurezza AWS, la risposta include le credenziali di sicurezza temporanee e un token di sessione. Il valore del token di sessione deve essere fornito nell'intestazione `x-amz-security-token` durante

l'invio di richieste ad Amazon S3. Per informazioni sull' Servizio di token di sicurezza AWS API fornita da IAM, vai a [Action](#) in the Servizio di token di sicurezza AWS API Reference Guide.

Intestazione di autenticazione

L'API REST di Amazon S3 utilizza l'intestazione standard HTTP `Authorization` per passare le informazioni di autenticazione (il nome dell'intestazione standard è poco felice perché in realtà comunica informazioni sull'autenticazione, non sull'autorizzazione). Nell'ambito dello schema di autenticazione Amazon S3, l'intestazione `Authorization` ha il formato seguente:

```
Authorization: AWS AWSAccessKeyId:Signature
```

Agli sviluppatori vengono rilasciati un ID della chiave di AWS accesso e una chiave di accesso AWS segreta al momento della registrazione. Per l'autenticazione delle richieste, l'elemento `AWSAccessKeyId` identifica l'ID chiave di accesso utilizzata per calcolare la firma e, indirettamente, lo sviluppatore che invia la richiesta.

L'`Signature` elemento è la RFC 2104 HMAC-SHA1 degli elementi selezionati dalla richiesta, pertanto la `Signature` parte dell'intestazione di autorizzazione varierà da richiesta a richiesta. Se la firma della richiesta calcolata dal sistema corrisponde a quella `Signature` inclusa nella richiesta, il richiedente avrà dimostrato il possesso della AWS chiave di accesso segreta. La richiesta verrà quindi elaborata con l'identità e l'autorità dello sviluppatore titolare della chiave.

Di seguito è riportato un esempio di grammatica che illustra la costruzione dell'intestazione `Authorization` per la richiesta. (Nell'esempio, l'elemento `\n` rappresenta il punto di codice Unicode `U+000A`, comunemente chiamato newline).

```
Authorization = "AWS" + " " + AWSAccessKeyId + ":" + Signature;

Signature = Base64( HMAC-SHA1( UTF-8-Encoding-Of(YourSecretAccessKey), UTF-8-Encoding-Of( StringToSign ) ) );

StringToSign = HTTP-Verb + "\n" +
  Content-MD5 + "\n" +
  Content-Type + "\n" +
  Date + "\n" +
  CanonicalizedAmzHeaders +
  CanonicalizedResource;
```

```
CanonicalizedResource = [ "/" + Bucket ] +  
    <HTTP-Request-URI, from the protocol name up to the query string> +  
    [ subresource, if present. For example "?acl", "?location", or "?logging"];  
  
CanonicalizedAmzHeaders = <described below>
```

HMAC-SHA1 è un algoritmo definito da [RFC 2104, Keyed-Hashing per l'autenticazione dei messaggi](#). L'algoritmo prende come input due stringhe in bit, una chiave e un messaggio. Per l'autenticazione delle richieste di Amazon S3, utilizza la tua chiave di accesso AWS segreta (YourSecretAccessKey) come chiave e la UTF-8 codifica come messaggio. StringToSign
L'output di HMAC-SHA1 è anche una stringa di byte, chiamata digest. Il parametro di richiesta Signature viene costruito da Base64 codificando il digest.

Standardizzazione della richiesta per la firma

È importante ricordare che, quando il sistema riceve una richiesta di autenticazione, confronta la firma calcolata con la firma fornita nella richiesta in StringToSign. Per questo motivo, devi calcolare la firma con lo stesso metodo utilizzato da Amazon S3. Il processo di presentazione di una richiesta in una forma concordata per la firma della canonicalizzazione.

Costruzione dell'elemento CanonicalizedResource

CanonicalizedResource rappresenta la risorsa Amazon S3 cui è destinata la richiesta. È possibile costruire l'elemento per una richiesta REST nel modo seguente:

1. Iniziare con una stringa vuota ("").
2. Se la richiesta specifica un bucket tramite l'intestazione Host HTTP (in stile hosting virtuale), aggiungere il nome del bucket preceduto da "/" (ad es., "/nomebucket"). Per le richieste in stile percorso o senza un bucket specifico, non è necessaria alcuna azione. Per ulteriori informazioni sulle richieste in stile host virtuale, vedere Hosting [virtuale di bucket](#).

Per una richiesta in stile host virtuale "«, è «https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg/awsexamplebucket1". CanonicalizedResource

Per la richiesta in stile percorso, "«, è «». https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg CanonicalizedResource

3. Aggiungi la parte del percorso dell'HTTP non decodificato Request-URI, fino alla stringa di query ma non inclusa.

Per una richiesta in stile host virtuale "«, è `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg` «`//puppy.jpg`». CanonicalizedResource `awsexamplebucket1/photos`

Per una richiesta in stile percorso, "`https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg`«, CanonicalizedResource è `«//puppy.jpg`». `awsexamplebucket1/photos` A questo punto, l'elemento CanonicalizedResource è lo stesso sia per le richieste in stile hosting virtuale che in stile percorso.

Per una richiesta non indirizzata a un bucket, come [GET Service](#), aggiungere `"/`.

4. Se la richiesta è indirizzata a una risorsa secondaria, come `?versioning`, `?location`, `?acl`, `?lifecycle` o `?versionid`, aggiungere la risorsa secondaria, il suo valore (se disponibile) e il punto interrogativo. Nota che in caso di più sottorisorse, le sottorisorse devono essere ordinate lessicograficamente in base al nome della sottorisorsa e separate da `'&'`, ad esempio `ACL&versionId=`. *value*

Le risorse secondarie da includere nella costruzione dell' CanonicalizedResource elemento sono `acl`, `lifecycle`, `location`, `logging`, `notification`, `partNumber`, `policy`, `requestPayment`, `uploadID`, `uploads`, `versionId`, `versionId`, `versioning`, `versions` e `website`.

Se la richiesta specifica parametri della stringa di query che sostituiscono i valori dell'intestazione della risposta (consultare [Get Object](#)), aggiungere i parametri della stringa di query e i rispettivi valori. Al momento della firma questi valori non vanno codificati, ma i valori dei parametri devono essere codificati quando si invia la richiesta. I parametri della stringa di query in una richiesta GET includono `response-content-type`, `response-content-language`, `response-expires`, `response-cache-control`, `response-content-disposition` e `response-content-encoding`.

Il parametro della stringa di `delete` query deve essere incluso quando si crea una richiesta di eliminazione per più oggetti. CanonicalizedResource

Gli elementi provenienti da HTTP Request-URI devono essere firmati letteralmente così come appaiono nella richiesta HTTP, inclusi i URL-Encoding metacaratteri. CanonicalizedResource

CanonicalizedResourcePotrebbe essere diverso dall'HTTP Request-URI. In particolare, se la richiesta utilizza l'Hostintestazione HTTP per specificare un bucket, il bucket non viene visualizzato nell'HTTP. Request-URI Tuttavia, nella CanonicalizedResource è sempre incluso il bucket. I

parametri della stringa di query potrebbero essere visualizzati anche in Request-URI ma non sono inclusi in `CanonicalizedResource`. Per ulteriori informazioni, consulta Hosting [virtuale di bucket](#).

Costruire l'elemento `CanonicalizedAmzHeaders`

Per creare la `CanonicalizedAmzHeaders` parte di `StringToSign`, seleziona tutte le intestazioni delle richieste HTTP che iniziano con 'x-amz-' (utilizzando un confronto senza distinzione tra maiuscole e minuscole) e usa la seguente procedura.

1. Convertire ciascun nome dell'intestazione HTTP in lettere minuscole. Ad esempio "X-Amz-Date" diventa "x-amz-date".
2. Disporre la raccolta delle intestazioni in ordine lessicografico in base al nome dell'intestazione.
3. Combinare i campi di intestazione con lo stesso nome in una coppia "nome-valore:elenco-valori-separati-da-virgole", come stabilito dallo standard RFC 2616, sezione 4.2, senza spazi tra i valori. Ad esempio, le due intestazioni con metadata "x-amz-meta-username: fred" e "x-amz-meta-username: barney" verrebbero combinate nell'intestazione singola "x-amz-meta-username: fred,barney".
4. "Aprire" le intestazioni più lunghe che occupano più righe (come consentito dallo standard RFC 2616, sezione 4.2) sostituendo gli spazi di folding (newline compreso) con un singolo spazio.
5. Rimuovere gli spazi attorno ai due punti nell'intestazione. Ad esempio, l'intestazione "diventerebbe". `x-amz-meta-username: fred,barney x-amz-meta-username: fred,barney`
6. Infine, aggiungere un carattere newline (U+000A) a ogni intestazione standardizzata nell'elenco risultante. Costruisci l' `CanonicalizedResource` elemento concatenando tutte le intestazioni di questo elenco in un'unica stringa.

Elementi di intestazione HTTP posizionali e denominati `StringToSign`

I primi elementi di intestazione di `StringToSign` (`Content-Type`, `Date` e `Content-MD5`) sono di natura posizionale. `StringToSign` non include i nomi di queste intestazioni, ma solo i valori relativi alla richiesta. Al contrario, gli elementi "x-amz-" sono denominati. In sono inclusi sia i nomi, sia i valori dell'intestazione `StringToSign`.

Se un'intestazione posizionale richiamata per la definizione di `StringToSign` non è presente nella richiesta (ad esempio, `Content-Type` o `Content-MD5` sono facoltative per le richieste PUT e prive di senso per le richieste GET), la stringa vuota ("") per tale posizione va sostituita.

Necessità del time stamp

Per le richieste autenticate, è obbligatorio un time stamp valido (tramite intestazione `Date` HTTP o un'alternativa `x-amz-date`). Inoltre, alla ricezione di una richiesta autenticata, il timestamp del client incluso nella richiesta non deve differire di oltre 15 minuti rispetto all'orario di sistema in Amazon S3. In caso contrario, la richiesta avrà esito negativo con il codice di errore `RequestTimeTooSkewed`. Tali restrizioni hanno lo scopo di limitare le possibilità che una richiesta intercettata possa essere riproposta da un sistema avversario. Per una maggiore protezione contro le intercettazioni, utilizzare il protocollo di trasferimento HTTPS per le richieste autenticate.

Note

Il vincolo relativo alla convalida della data della richiesta è valido solo per le richieste autenticate che non utilizzano l'autenticazione stringa di query. Per ulteriori informazioni, consulta [Alternativa per l'autenticazione di una richiesta tramite stringa di query](#).

Alcune librerie client HTTP non danno la possibilità di impostare l'intestazione `Date` per una richiesta. Se includere il valore dell'intestazione `"Date"` nelle intestazioni standardizzate crea difficoltà, è possibile impostare il time stamp per la richiesta utilizzando l'intestazione `"x-amz-date"`. Il valore dell'intestazione `x-amz-date` deve essere in uno dei formati RFC 2616 (<http://www.ietf.org/rfc/rfc2616.txt>). Quando in una richiesta è presente un'intestazione `x-amz-date`, durante il calcolo della firma della richiesta il sistema ignorerà qualsiasi intestazione di tipo `Date`. Di conseguenza, se si include l'intestazione `x-amz-date`, va utilizzata la stringa vuota per il valore `Date` durante la costruzione della stringa `StringToSign`. Nella prossima sezione sarà riportato un esempio.

Esempi di autenticazione

Gli esempi in questa sezione utilizzano le credenziali (non funzionanti) incluse nella seguente tabella.

Parametro	Valore
<code>AWSAccessKeyId</code>	<code>AKIAIOSFODNN7EXAMPLE</code>
<code>AWSSecretAccessKey</code>	<code>wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY</code>

Negli elementi `StringToSign` di esempio, la formattazione non è significativa, mentre l'elemento `\n` rappresenta il punto di codice Unicode U+000A, comunemente chiamato `newline`. Inoltre, negli esempi viene utilizzato "+0000" per identificare il fuso orario. Allo stesso scopo è anche possibile utilizzare "GMT", ma le firme mostrate nell'esempio saranno diverse.

Richiesta GET di un oggetto

Nell'esempio viene richiesto un oggetto dal bucket `awsexamplebucket1`.

Richiesta	StringToSign
<pre>GET /photos/puppy.jpg HTTP/1.1 Host: awsexamplebucket1.us- west-1.s3.amazonaws.com Date: Tue, 27 Mar 2007 19:36:42 +0000 Authorization: AWS AKIAIOSF0 DNN7EXAMPLE: qgk2+6Sv9/oM7G3qLEjTH1a1l1g=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:36:42 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

Nota che `CanonicalizedResource` include il nome del bucket, ma l'HTTP Request-URI no. (Il bucket è specificato dall'intestazione `Host`).

Note

Il seguente script Python calcola la firma precedente utilizzando i parametri forniti. Puoi usare questo script per creare le tue firme, sostituendo le chiavi e `StringToSign` se necessario.

```
import base64
import hmac
from hashlib import sha1

access_key = 'AKIAIOSF0DNN7EXAMPLE'.encode("UTF-8")
secret_key = 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'.encode("UTF-8")

string_to_sign = 'GET\n\n\nTue, 27 Mar 2007 19:36:42 +0000\n/awsexamplebucket1/
photos/puppy.jpg'.encode("UTF-8")
signature = base64.b64encode(
```

```

        hmac.new(
            secret_key, string_to_sign, sha1
        ).digest()
    ).strip()

print(f"AWS {access_key.decode()}:{signature.decode()}")

```

Richiesta PUT di un oggetto

Nell'esempio viene inserito un oggetto nel bucket `awsexamplebucket1`.

Richiesta	StringToSign
<pre> PUT /photos/puppy.jpg HTTP/1.1 Content-Type: image/jpeg Content-Length: 94328 Host: awsexamplebucket1.s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 21:15:45 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: iqRzw+ileNPu1fhspnRs8n0jjIA= </pre>	<pre> PUT\n \n image/jpeg\n Tue, 27 Mar 2007 21:15:45 +0000\n /awsexamplebucket1/photos/puppy.jpg </pre>

Annota l' Content-Type intestazione nella richiesta e nel. StringToSign Nota inoltre che Content-MD5 è lasciato vuoto in StringToSign, perché non è presente nella richiesta.

List

Nell'esempio viene recuperato l'elenco dei contenuti del bucket `awsexamplebucket1`.

Richiesta	StringToSign
<pre> GET /?prefix=photos&max-keys=50&marker=puppy HTTP/1.1 User-Agent: Mozilla/5.0 </pre>	<pre> GET\n \n \n </pre>

Richiesta	StringToSign
<pre>Host: awsexamplebucket1.s3.us-west-1.amazo naws.com Date: Tue, 27 Mar 2007 19:42:41 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: m0WP8eCtspQl5Ahe6L1SozdX9YA=</pre>	<pre>Tue, 27 Mar 2007 19:42:41 +0000\n /awsexamplebucket1/</pre>

Nota la barra finale CanonicalizedResource e l'assenza di parametri della stringa di query.

Recupero

Nell'esempio viene recuperata la risorsa secondaria relativa alla policy di controllo degli accessi per il bucket "awsexamplebucket1".

Richiesta	StringToSign
<pre>GET /?acl HTTP/1.1 Host: awsexamplebucket1.s3.us-west-1.amazo naws.com Date: Tue, 27 Mar 2007 19:44:46 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: 82ZHiFIjc+WbcwFKGUVeqspPn+0=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:44:46 +0000\n /awsexamplebucket1/?acl</pre>

Notate come il parametro della stringa di interrogazione della subrisorsa sia incluso in CanonicalizedResource

Elimina

In questo esempio viene eliminato un oggetto dal bucket "awsexamplebucket1" utilizzando lo stile percorso e l'alternativa Date.

Richiesta	StringToSign
<pre>DELETE /awsexamplebucket1/photos/p uppy.jpg HTTP/1.1</pre>	<pre>DELETE\n \n</pre>

Richiesta	StringToSign
<pre>User-Agent: dotnet Host: s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 21:20:27 +0000 x-amz-date: Tue, 27 Mar 2007 21:20:26 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: XbyTlbQdu9Xw5o8P4iMwPktxQd8=</pre>	<pre>\n Tue, 27 Mar 2007 21:20:26 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

Nota il metodo alternativo 'x-amz-date' per specificare la data (perché la nostra libreria client ci ha impedito di impostare la data, ad esempio). In questo caso, il metodo x-amz-date prevale sull'intestazione Date. Di conseguenza, la voce relativa alla data nella firma deve contenere il valore dell'intestazione x-amz-date.

Caricamento

In questo esempio viene caricato un oggetto in un bucket CNAME in stile hosting virtuale con metadata.

Richiesta	StringToSign
<pre>PUT /db-backup.dat.gz HTTP/1.1 User-Agent: curl/7.15.5 Host: static.example.com:8080 Date: Tue, 27 Mar 2007 21:06:08 +0000 x-amz-acl: public-read content-type: application/x-download Content-MD5: 4gJE4saaMU4BqNR0kLY+lw== X-Amz-Meta-ReviewedBy: joe@example.com X-Amz-Meta-ReviewedBy: jane@example.com X-Amz-Meta-FileChecksum: 0x02661779 X-Amz-Meta-ChecksumAlgorithm: crc32 Content-Disposition: attachment; filename=database.dat Content-Encoding: gzip Content-Length: 5913339</pre>	<pre>PUT\n 4gJE4saaMU4BqNR0kLY+lw==\n application/x-download\n Tue, 27 Mar 2007 21:06:08 +0000\n x-amz-acl:public-read\n x-amz-meta-checksumalgorithm:crc32\n x-amz-meta-filechecksum:0x02661779\n x-amz-meta-reviewedby:joe@example.com,jane@example.com\n /static.example.com/db-backup.dat.gz</pre>

Richiesta	StringToSign
<i>Authorization: AWS AKIAIOSFODNN7EXAMPLE LE: jtBQa0Aq+DkULFI8qrpwIjGEx0E=</i>	

Osserva il modo in cui le intestazioni "x-amz-" vengono ordinate, private degli spazi aggiuntivi e convertite in caratteri minuscoli. Inoltre, più intestazioni con lo stesso nome sono state unite, con i valori separati da virgole.

Si noti come solo le intestazioni delle entità HTTP Content-Type e Content-MD5 siano presenti in StringToSign, mentre le intestazioni dell'altra entità Content-* non lo sono.

Ancora una volta, nota che CanonicalizedResource include il nome del bucket, ma l'HTTP no. Request-URI (Il bucket è specificato dall'intestazione Host).

Elenco di tutti i bucket personali

Richiesta	StringToSign
GET / HTTP/1.1 Host: s3.us-west-1.amazonaws.com Date: Wed, 28 Mar 2007 01:29:59 +0000 <i>Authorization: AWS AKIAIOSFODNN7EXAMPLE:qGdzdE RIC03wnaRNKh60qZehG9s=</i>	GET\n \n \n Wed, 28 Mar 2007 01:29:59 +0000\n /

Chiavi Unicode

Richiesta	StringToSign
GET /dictionary/fran%C3%A7ais/pr %c3%a9f%c3%a8re HTTP/1.1 Host: s3.us-west-1.amazonaws.com Date: Wed, 28 Mar 2007 01:49:49 +0000 <i>Authorization: AWS AKIAIOSFODNN7EXAMPLE LE:DNEZGsoieTZ92F3bUfSPQcbGm1M=</i>	GET\n \n \n Wed, 28 Mar 2007 01:49:49 +0000\n /dictionary/fran%C3%A7ais/pr %c3%a9f%c3%a8re

 Note

Gli elementi `StringToSign` che ne derivano `Request-URI` sono presi alla lettera, compresi quelli URL-Encoding in maiuscolo.

Problemi con la firma delle richieste REST

In caso di errore nell'autenticazione delle richieste REST, il sistema risponde alla richiesta con un documento di errore XML, che contiene informazioni concepite per aiutare gli sviluppatori a individuare il problema. In particolare, l'elemento `StringToSign` del documento di errore `SignatureDoesNotMatch` indica esplicitamente il tipo di standardizzazione della richiesta usata dal sistema.

Alcuni kit di strumenti inseriscono tacitamente intestazioni non note in precedenza, ad esempio nel caso dell'aggiunta dell'intestazione `Content-Type` durante un'operazione `PUT`. In gran parte dei casi, il valore dell'intestazione inserita rimane costante: questo consente di scoprire le intestazioni mancanti utilizzando strumenti come `Ethereal` o `tcpmon`.

Alternativa per l'autenticazione di una richiesta tramite stringa di query

È possibile autenticare alcuni tipi di richieste passando le informazioni necessarie come parametri di una stringa di query, invece di utilizzare l'intestazione `HTTP Authorization`. Questa soluzione è utile per abilitare l'accesso diretto da parte di browser di terze parti ai dati Amazon S3 privati senza proxy della richiesta. L'idea è costruire di una richiesta "prefirmata" e codificarla come URL recuperabile da parte del browser di un utente finale. È inoltre possibile limitare una richiesta prefirmata specificando un periodo di scadenza.

Per ulteriori informazioni sull'utilizzo dei parametri di query per autenticare le richieste, consulta [Authenticating Requests: Using Query Parameters \(AWS Signature Version 4\)](#) nell'Amazon Simple Storage Service API Reference. Per esempi di utilizzo degli AWS SDK per generare URL prefirmiti, consulta [Condivisione di oggetti con URL prefirmiti](#).

Creazione di una firma

Di seguito è riportato un esempio di richiesta REST Amazon S3 autenticata tramite stringa di query.

```
GET /photos/puppy.jpg
?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Expires=1141889120&Signature=vjbyPxybdZaNmGa%2ByT272YEAiv4%3D HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
Date: Mon, 26 Mar 2007 19:37:58 +0000
```

Il metodo di autenticazione delle richieste tramite stringa di query non richiede particolari intestazioni HTTP. Gli elementi necessari all'autenticazione sono invece specificati come parametri della stringa di query:

Nome parametro stringa di query	Valore di esempio	Description
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE	L'ID della tua chiave di accesso. AWS Specifica la chiave di accesso AWS segreta utilizzata per firmare la richiesta e, indirettamente, l'identità dello sviluppatore che effettua la richiesta.
Expires	1141889120	L'ora di scadenza della firma, specificata come numero di secondi dal valore epoca (Unix epoch, ovvero 00:00:00 UTC del 1° gennaio 1970). Se ricevuta dopo l'ora indicata (secondo il server), la richiesta verrà rifiutata.
Signature	vjbyPxybdZaNmGa%2ByT272YEAiv4%3D	La codifica URL della codifica Base64 di of. HMAC-SHA1 StringToSign

Il metodo di autenticazione delle richieste tramite stringa di query varia leggermente rispetto al metodo tradizionale, ma solo nel formato del parametro di richiesta `Signature` e nell'elemento `StringToSign`. Di seguito è riportato un esempio di grammatica che illustra il metodo di autenticazione delle richieste tramite stringa di query.

```
Signature = URL-Encode( Base64( HMAC-SHA1( YourSecretAccessKey, UTF-8-Encoding-Of( StringToSign ) ) ) );
```

```
StringToSign = HTTP-VERB + "\n" +
    Content-MD5 + "\n" +
    Content-Type + "\n" +
    Expires + "\n" +
    CanonicalizedAmzHeaders +
    CanonicalizedResource;
```

`YourSecretAccessKey` è l'ID della chiave di accesso AWS segreta che Amazon ti assegna quando ti registri per diventare uno sviluppatore di Amazon Web Service. Nota come `Signature` è URL-Encoded necessario renderlo adatto per il posizionamento nella stringa di query. Inoltre, in `StringToSign`, l'elemento posizionale `HTTP Date` è stato sostituito con l'elemento `Expires`, mentre `CanonicalizedAmzHeaders` e `CanonicalizedResource` restano invariati.

Note

Nel metodo di autenticazione stringa di query, non si utilizzano né l'elemento `Date` né l'intestazione `x-amz-date request` per il calcolo della stringa da firmare.

Autenticazione di una richiesta tramite stringa di query

Richiesta	StringToSign
<pre>GET /photos/puppy.jpg?AWSAccess KeyId=AKIAIOSFODNN7EXAMPLE& Signature=NpgCjnDzrM%2BWFzo ENXmpNDUsSn8%3D& Expires=1175139620 HTTP/1.1 Host: awsexamplebucket1.s3.us-wes t-1.amazonaws.com</pre>	<pre>GET\n \n \n 1175139620\n /awsexamplebucket1/photos/puppy.jpg</pre>

Questo esempio presuppone che quando un browser effettua la richiesta GET, non fornisca `Content-MD5` né un' `Content-Type` intestazione né imposti alcuna intestazione `x-amz-`, quindi tali parti vengono lasciate vuote. `StringToSign`

Utilizzo della codifica Base64

Le firme di richiesta HMAC devono essere codificate in formato Base64. La codifica Base64 converte la firma in una stringa ASCII semplice che è possibile allegare alla richiesta. Se utilizzati in un URI, i caratteri che possono essere presenti in una firma, ad esempio il più (+), la barra (/) e l'uguale (=), devono essere codificati. Ad esempio, se il codice di autenticazione include un simbolo più (+), nella richiesta va codificato come %2B. La codifica per la barra è %2F, mentre per l'uguale è %3D.

Per esempi di codifica Base64, consulta di Amazon 3 [Esempi di autenticazione](#).

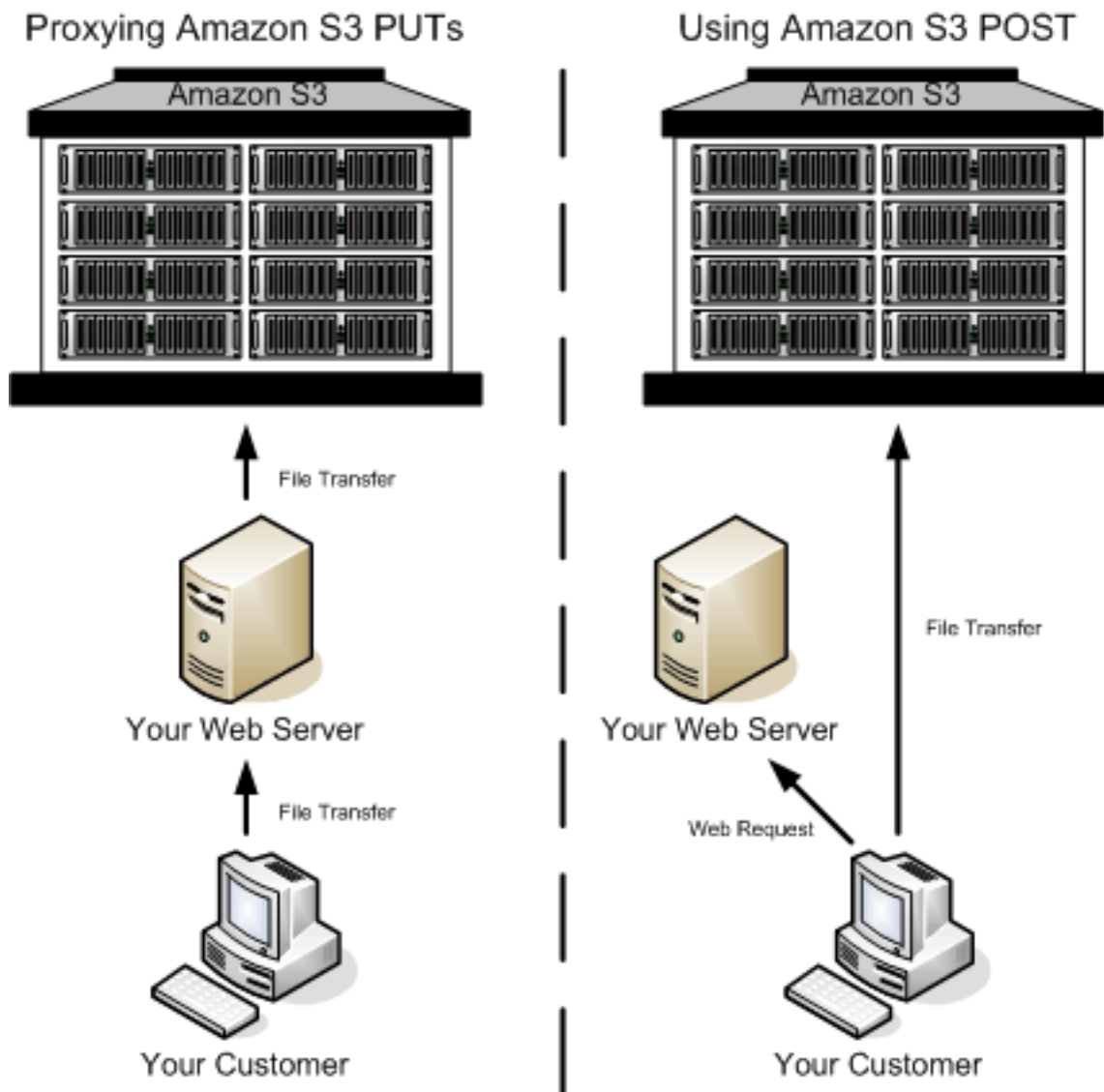
Browser-based caricamenti utilizzando POST (AWS firma (versione 2))

Amazon S3 supporta POST, che permette agli utenti di caricare contenuti direttamente in Amazon S3. POST è progettato per semplificare i caricamenti, ridurre la latenza e risparmiare denaro sulle applicazioni in cui gli utenti caricano dati da archiviare in Amazon S3.

Note

L'autenticazione delle richieste discussa in questa sezione si basa su AWS Signature Version 2, un protocollo per l'autenticazione delle richieste API in entrata ai servizi. AWS Amazon S3 ora supporta complessivamente la versione 4 di Signature, un protocollo per l'autenticazione delle richieste API in entrata ai servizi. AWS Regioni AWS Al momento, Regioni AWS creato prima del 30 gennaio 2014, continuerà a supportare il protocollo precedente, Signature Version 2. Le regioni create dopo il 30 gennaio 2014 supporteranno solo Signature Version 4 e pertanto tutte le richieste a tali regioni devono essere effettuate con Signature Version 4. Per ulteriori informazioni, consulta [Autenticazione delle richieste nei Browser-Based caricamenti tramite POST \(AWS Signature Version 4\)](#) nel Amazon Simple Storage Service API Reference.

La figura seguente mostra un caricamento con POST in Amazon S3.



Caricamento tramite POST

- 1 L'utente apre un browser Web e accede alla tua pagina Web.
- 2 La tua pagina Web contiene un modulo HTTP con tutte le informazioni necessarie all'utente per caricare contenuti in Amazon S3.
- 3 L'utente carica contenuti direttamente in Amazon S3.

Note

L'autenticazione stringa di query non è supportata per POST.

Moduli HTML (AWS firma (versione 2))

Argomenti

- [Codifica dei moduli HTML](#)
- [Dichiarazione del modulo HTML](#)
- [Campi del modulo HTML](#)
- [Costruzione della policy](#)
- [Costruzione di una firma](#)
- [Reindirizzamento](#)

Quando comunichi con Amazon S3, utilizzi in genere l'API REST o SOAP per eseguire operazioni di inserimento, recupero e di altro tipo. Con POST, gli utenti caricano i dati direttamente in Amazon S3 tramite i propri browser, che non possono elaborare l'API SOAP o creare una richiesta REST PUT.

Note

Le API SOAP per Amazon S3 non sono disponibili per i nuovi clienti, con la fine del ciclo di vita prevista per il 31 agosto 2025. Ti consigliamo di utilizzare l'API REST o gli AWS SDK.

Per permettere agli utenti di caricare contenuti in Amazon S3 utilizzando i propri browser, devi utilizzare moduli HTML. I moduli HTML si compongono di una dichiarazione del modulo e di campi del modulo. La dichiarazione del modulo contiene informazioni generali sulla richiesta. I campi del modulo contengono informazioni dettagliate sulla richiesta, nonché la policy utilizzata per autenticarla e garantire che soddisfi le condizioni specificate.

Note

I dati e i limiti del modulo (esclusi i contenuti del file) non possono superare 20 KB.

Questa sezione spiega come utilizzare i moduli HTML.

Codifica dei moduli HTML

Il modulo e la policy devono essere UTF-8 codificati. È possibile applicare la UTF-8 codifica al modulo specificandolo nell'intestazione HTML o come intestazione della richiesta.

 Note

La dichiarazione del modulo HTML non accetta parametri di autenticazione stringa di query.

Di seguito è riportato un esempio di UTF-8 codifica nell'intestazione HTML:

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

Di seguito è riportato un esempio di UTF-8 codifica nell'intestazione di una richiesta:

```
Content-Type: text/html; charset=UTF-8
```

Dichiarazione del modulo HTML

La dichiarazione del modulo presenta tre componenti: l'operazione, il metodo e il tipo di inquadramento. Se uno di questi valori non è impostato correttamente, la richiesta ha esito negativo.

L'operazione specifica l'URL che elabora la richiesta, il quale deve essere impostato sull'URL del bucket. Ad esempio, se il nome del bucket è `awsexamplebucket1` e la regione è Stati Uniti occidentali (California settentrionale), l'URL è `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/`.

 Note

Il nome della chiave è specificato nel campo del modulo.

Il metodo deve essere POST.

Il tipo di allegato (enctype) deve essere specificato e deve essere impostato su multipart/form-data sia per i caricamenti di file che per i caricamenti di aree di testo. Per ulteriori informazioni, consulta [RFC 1867](#).

Example

L'esempio seguente è la dichiarazione di un modulo per il bucket "awsexamplebucket".

```
<form action="https://awsexamplebucket1.s3.us-west-1.amazonaws.com/" method="post"
enctype="multipart/form-data">
```

Campi del modulo HTML

Nella seguente tabella sono descritti i campi che possono essere utilizzati in un modulo HTML.

Note

La variabile `${filename}` viene sostituita automaticamente con il nome del file fornito dall'utente ed è riconosciuta da tutti i campi del modulo. Se il browser o il client fornisce un percorso completo o parziale al file, verrà utilizzato solo il testo che segue l'ultima barra (/) o barra rovesciata (\). Ad esempio, "C:\Program Files\directory1\file.txt" viene interpretato come "file.txt". Se non vengono forniti file o nomi di file, la variabile viene sostituita con una stringa vuota.

Nome campo	Descrizione	Richiesto
AWSAccessKeyId	L'ID della chiave di AWS accesso del proprietario del bucket che concede a un utente anonimo l'accesso a una richiesta che soddisfa il set di vincoli della policy. Questo campo è obbligatorio se la richiesta include un documento di policy.	Condizionale
acl	Lista di controllo accessi (ACL) Amazon S3. Se si specifica una lista di controllo accessi non valida, viene generato un errore.	No

Nome campo	Descrizione	Richiesto
	Tipo: stringa Default: private Valori validi: private public-read public-read-write aws-exec-read authenticated-read bucket-owner-read bucket-owner-full-control	
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	REST-specific intestazioni. Per ulteriori informazioni, consulta PUT Object .	No
key	Il nome della chiave caricata. Per utilizzare il nome del file fornito dall'utente, utilizzare la variabile <code>\${filename}</code>. Ad esempio, se l'utente Betty carica il file lolcatz.jpg e si specifica <code>user/betty/\${filename}</code>, il file viene archiviato come <code>//lolcatz.jpg. user/betty</code> Per ulteriori informazioni, consulta Utilizzo dei metadati degli oggetti.	Sì
policy	Policy di sicurezza che descrive cosa è permesso nella richiesta. Le richieste senza policy di sicurezza sono considerate anonime e hanno esito positivo solo su bucket pubblicamente scrivibili.	No

Nome campo	Descrizione	Richiesto
<code>success_action_redirect, redirect</code>	L'URL a cui viene reindirizzato il client dopo il corretto caricamento. Amazon S3 aggiunge i valori di bucket, chiave ed ETag come parametri della stringa di query all'URL. Se <code>success_action_redirect</code> non è specificato, Amazon S3 restituisce il tipo di documento vuoto specificato nel campo <code>success_action_status</code>. Se Amazon S3 non riesce a interpretare l'URL, ignora il campo . Se il caricamento non riesce, Amazon S3 visualizza un errore e non reindirizza l'utente a un URL. Per ulteriori informazioni, consulta Reindirizzamento.  Note Il nome del campo di reindirizzamento è obsoleto e in futuro il supporto del nome del campo di reindirizzamento verrà eliminato.	No

Nome campo	Descrizione	Richiesto
success_action_status	Il codice di stato restituito al client dopo il corretto caricamento, se non è specificato success_action_redirect. I valori validi sono 200, 201 o 204 (default). Se il valore è impostato su 200 o 204, Amazon S3 restituisce un documento vuoto con codice di stato 200 o 204. Se il valore è impostato su 201, Amazon S3 restituisce un documento XML con codice di stato 201. Per informazioni sul contenuto del documento XML, consulta POST Object. Se il valore non è impostato o non è valido, Amazon S3 restituisce un documento vuoto con codice di stato 204.	No

 **Note**

Alcune versioni di Adobe Flash Player non gestiscono correttamente le risposte HTTP con corpo vuoto. Per supportare i caricamenti tramite Adobe Flash, si consiglia di impostare success_action_status su 201.

Nome campo	Descrizione	Richiesto
signature	La firma HMAC costruita utilizzando la chiave di accesso segreta che corrisponde a quella fornita. <code>AWSAccessKeyId</code> Questo campo è obbligatorio se la richiesta include un documento di policy. Per ulteriori informazioni, consulta Identity and Access Management for Amazon S3.	Condizionale
x-amz-security-token	Token di sicurezza utilizzato dalle credenziali per la sessione Se la richiesta utilizza Amazon DevPay , sono necessari due campi del <code>x-amz-security-token</code> modulo: uno per il token del prodotto e uno per il token utente. Se la richiesta utilizza le credenziali per la sessione, richiede un modulo <code>x-amz-security-token</code> . Per ulteriori informazioni, consulta Credenziali di sicurezza temporanee nella Guida per l'utente di IAM.	No
Altri nomi di campi con prefisso x-amz-meta-	User-specified metadati. Amazon S3 non convalida o utilizza questi dati. Per ulteriori informazioni, consulta PutObject.	No

Nome campo	Descrizione	Richiesto
file	File o contenuto di testo. Il file o il contenuto deve essere l'ultimo campo del modulo. Tutti i campi al di sotto di questo vengono ignorati. Non è possibile caricare più di un file alla volta.	Sì

Costruzione della policy

Argomenti

- [Scadenza](#)
- [Condizioni](#)
- [Corrispondenza delle condizioni](#)
- [Utilizzo di escape con caratteri](#)

La policy è un UTF-8 documento Base64-encoded JSON che specifica le condizioni che la richiesta deve soddisfare e viene utilizzato per autenticare il contenuto. A seconda di come si progettano i documenti di policy, è possibile utilizzarli per caricamento, per utente, per tutti i caricamenti o secondo altre progettazioni in base alle esigenze.

Note

Sebbene il documento di policy sia facoltativo, è fortemente consigliato rispetto a rendere il bucket pubblicamente scrivibile.

Di seguito è illustrato un esempio di documento di policy:

```
{ "expiration": "2007-12-01T12:00:00.000Z",  
  
  "conditions": [
```

```
{ "acl": "public-read" },  
  
  { "bucket": "awsexamplebucket1" },  
  
  [ "starts-with", "$key", "user/eric/" ],  
  
]  
  
}
```

Il documento di policy contiene la scadenza e le condizioni.

Scadenza

L'elemento scadenza specifica la data di scadenza della policy in formato ISO 8601 UTC. Ad esempio, «2007-12-01T12:00:00.000Z» specifica che la politica non è valida dopo la mezzanotte UTC del 01/12/2007. In una policy la scadenza è obbligatoria.

Condizioni

Le condizioni nel documento di policy convalidano i contenuti dell'oggetto caricato. Ogni campo del modulo specificato nel modulo (ad eccezione della firma `AWSAccessKeyId`, del file, della politica e dei nomi di campo con il prefisso `x-ignore-`) deve essere incluso nell'elenco delle condizioni.

Note

Se si dispone di più campi con lo stesso nome, i valori devono essere separati da virgole. Ad esempio, se si dispone di due campi denominati `"x-amz-meta-tag"`, dove il valore del primo è `"Ninja"` e quello del secondo è `"Stallman"`, si imposta il documento di policy su `Ninja,Stallman`.

Tutte le variabili all'interno del modulo vengono estese prima della convalida della policy. Pertanto, tutte le corrispondenze delle condizioni devono essere eseguite rispetto ai campi estesi. Ad esempio, se si imposta il campo della chiave su `user/betty/${filename}`, la policy potrebbe essere `[ "starts-with", "$key", "user/betty/" ]`. Non inserire `[ "starts-with", "$key", "user/betty/${filename}" ]`. Per ulteriori informazioni, consulta [Corrispondenza delle condizioni](#).

La tabella seguente descrive le condizioni del documento di policy.

Nome elemento	Descrizione
acl	Specifica le condizioni che l'ACL deve soddisfare. Supporta la corrispondenza esatta e <code>starts-with</code> .
content-length-range	Specifica la dimensione minima e massima consentita per il contenuto caricato. Supporta la corrispondenza dell'intervallo.
Cache-Control,, Content-Type, Content-Disposition, Scade Content-Encoding	REST-specific intestazioni. Supporta la corrispondenza esatta e <code>starts-with</code> .
key	Il nome della chiave caricata. Supporta la corrispondenza esatta e <code>starts-with</code> .
success_action_redirect, redirect	L'URL a cui viene reindirizzato il client dopo il corretto caricamento. Supporta la corrispondenza esatta e <code>starts-with</code> .
success_action_status	Il codice di stato restituito al client dopo il corretto caricamento, se non è specificato <code>success_action_redirect</code>. Supporta la corrispondenza esatta.
x-amz-security-token	Token DevPay di sicurezza Amazon. Ogni richiesta che utilizza Amazon DevPay richiede due campi del <code>x-amz-security-token</code> modulo: uno per il token del prodotto e uno per il token utente. Di conseguenza, i valori devono essere separati da virgole. Ad esempio, se il token utente è <code>eW91dHVIZQ==</code> e il token prodotto

Nome elemento	Descrizione
	è b0hnNVNKWVJIQTA= , si imposta la voce di policy su: { "x-amz-security-token": "ew91dHViZ Q==,b0hnNVNKWVJIQTA=" } .
Altri nomi di campi con prefisso x-amz-meta-	User-specified metadata. Supporta la corrispondenza esatta e starts-with .

Note

Se il kit di strumenti aggiunge ulteriori campi (ad esempio, Flash aggiunge il nome del file), è necessario aggiungerli al documento di policy. Se puoi controllare questa funzionalità, aggiungi il prefisso `x-ignore-` al campo in modo che Amazon S3 ignori la funzionalità e non ne alteri le versioni future.

Corrispondenza delle condizioni

La tabella seguente descrive i tipi di corrispondenza delle condizioni. Sebbene occorra specificare una condizione per ciascun campo del modulo specificato nel modulo, è possibile creare criteri di corrispondenza più complessi specificando più condizioni per un campo del modulo.

Condition	Descrizione
Corrispondenze esatte	Le corrispondenze esatte verificano che i campi corrispondano a specifici valori. Questo esempio indica che l'ACL deve essere impostata su public-read: <pre>{"acl": "public-read" }</pre> Questo esempio rappresenta un modo alternativo per indicare che l'ACL deve essere impostata su public-read: <pre>["eq", "\$acl", "public-read"]</pre>

Condition	Descrizione
Inizia con	Se il valore deve iniziare con un determinato valore, utilizzare starts-with. Questo esempio indica che la chiave deve iniziare con: user/betty <pre>["starts-with", "\$key", "user/betty/"]</pre>
Corrispon- denza di qualsiasi contenuto	Per configurare la policy in modo da autorizzare qualsiasi contenuto in un campo, utilizzare starts-with con un valore vuoto. Questo esempio permette qualsiasi success_action_redirect: <pre>["starts-with", "\$success_action_redirect", ""]</pre>
Specifica degli intervalli	Per i campi che accettano intervalli, separare gli intervalli superiori e inferiori con una virgola. Questo esempio permette una dimensione dei file da 1 a 10 megabyte: <pre>["content-length-range", 1048579, 10485760]</pre>

Utilizzo di escape con caratteri

La tabella seguente descrive i caratteri da inserire in una sequenza di escape in un documento di policy.

Sequenza di escape	Descrizione
\\	Barra rovesciata
\\$	Simbolo del dollaro

Sequenza di escape	Descrizione
<code>\b</code>	Backspace
<code>\f</code>	Avanzamento modulo
<code>\n</code>	Nuova riga
<code>\r</code>	Ritorno a capo
<code>\t</code>	Tabulatore orizzontale
<code>\v</code>	Tabulatore verticale
<code>\ u xxxx</code>	Tutti i caratteri Unicode

Costruzione di una firma

Fase	Descrizione
1	Codificare la politica utilizzando UTF-8.
2	Codifica quei UTF-8 byte usando Base64.
3	Firma la policy con la tua chiave di accesso segreta utilizzando HMAC. SHA-1
4	Codifica la SHA-1 firma utilizzando Base64.

Per informazioni generali sull'autenticazione, consulta [Identity and Access Management for Amazon S3](#).

Reindirizzamento

Questa sezione descrive come gestire i reindirizzamenti.

Reindirizzamento generico

Al completamento della richiesta POST, l'utente viene reindirizzato alla posizione specificata nel campo `success_action_redirect`. Se Amazon S3 non riesce a interpretare l'URL, ignora il campo `success_action_redirect`.

Se `success_action_redirect` non è specificato, Amazon S3 restituisce il tipo di documento vuoto specificato nel campo `success_action_status`.

Se la richiesta POST non riesce, Amazon S3 visualizza un errore e non fornisce un reindirizzamento.

Pre-upload reindirizzamento

Se il bucket è stato creato utilizzando `< CreateBucketConfiguration >`, gli utenti finali potrebbero richiedere un reindirizzamento. In tal caso, alcuni browser potrebbero gestire il reindirizzamento in modo errato. Ciò è relativamente raro, ma vi è maggiore probabilità che si verifichi subito dopo la creazione del bucket.

Esempi di caricamento (AWS firma (versione 2))

Argomenti

- [Caricamento di un file](#)
- [Caricamento di un'area di testo](#)

Note

L'autenticazione delle richieste discussa in questa sezione si basa su AWS Signature Version 2, un protocollo per l'autenticazione delle richieste API in entrata ai servizi. AWS Amazon S3 ora supporta complessivamente la versione 4 di Signature, un protocollo per l'autenticazione delle richieste API in entrata ai servizi. AWS Regioni AWS Al momento, Regioni AWS creato prima del 30 gennaio 2014, continuerà a supportare il protocollo precedente, Signature Version 2. Le regioni create dopo il 30 gennaio 2014 supporteranno solo Signature Version 4 e pertanto tutte le richieste a tali regioni devono essere effettuate con Signature Version 4. Per ulteriori informazioni, consulta [Esempi: Browser-Based Upload](#)

[using HTTP POST \(Using AWS Signature Version 4\)](#) nell'Amazon Simple Storage Service API Reference.

Caricamento di un file

Questo esempio illustra il processo completo di costruzione di una policy e di un modulo da utilizzare per caricare un file allegato.

Costruzione di una policy e di un modulo

La policy seguente supporta caricamenti in Amazon S3 per il bucket `awsexamplebucket1`.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html"},
    ["starts-with", "$Content-Type", "image/"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

Questa policy richiede quanto segue:

- Il caricamento deve avvenire prima delle 12:00, orario UTC, del 1 dicembre 2007.
- Il contenuto deve essere caricato nel bucket `awsexamplebucket1`.
- La chiave deve iniziare con `"user/eric/»"`.
- L'ACL è impostata su `public-read`.
- Il `success_action_redirect` è impostato su `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html`.
- L'oggetto è un file di immagine.
- Il tag `x-amz-meta-uuid` deve essere impostato su `14365123651274`.
- Il campo `x-amz-meta-tag` può contenere qualsiasi valore.

Di seguito è riportata una versione di questa politica. Base64-encoded

eyJ0eXhwaXJhdGlubiI6ICIyMDA3LTExLTAxVDEyOjAwMjAwLjAwMFoiLAogICJjb25kaXRpb25zIjoqWwogICAgeyJidWN

Creare una firma utilizzando le credenziali personali. Ad esempio, 0RavWzkygo6QX9caELEqKi9kDbU= è la firma per il documento di policy precedente.

Il modulo seguente supporta una richiesta POST al amzn-s3-demo-bucket bucket che utilizza questa politica.

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
    ...
    <form action="https://amzn-s3-demo-bucket.s3.us-west-1.amazonaws.com/" method="post"
    enctype="multipart/form-data">
      Key to upload: <input type="input" name="key" value="user/eric/" /><br />
      <input type="hidden" name="acl" value="public-read" />
      <input type="hidden" name="success_action_redirect" value="https://
awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html" />
      Content-Type: <input type="input" name="Content-Type" value="image/jpeg" /><br />
      <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
      Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
      <input type="hidden" name="AWSAccessKeyId" value="AKIAIOSFODNN7EXAMPLE" />
      <input type="hidden" name="Policy" value="POLICY" />
      <input type="hidden" name="Signature" value="SIGNATURE" />
      File: <input type="file" name="file" /> <br />
      <!-- The elements after this will be ignored -->
      <input type="submit" name="submit" value="Upload to Amazon S3" />
    </form>
    ...
  </html>
```

Richiesta di esempio

Questa richiesta presuppone che l'immagine caricata abbia una dimensione di 117.108 byte; i dati dell'immagine non sono inclusi.

```
POST / HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10) Gecko/20071115
  Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=9431149156168
Content-Length: 118698

--9431149156168
Content-Disposition: form-data; name="key"

user/eric/MyPicture.jpg
--9431149156168
Content-Disposition: form-data; name="acl"

public-read
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"

https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html
--9431149156168
Content-Disposition: form-data; name="Content-Type"

image/jpeg
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-tag"

Some,Tag,For,Picture
--9431149156168
Content-Disposition: form-data; name="AWSAccessKeyId"

AKIAIOSFODNN7EXAMPLE
--9431149156168
```

```
Content-Disposition: form-data; name="Policy"

eyJAiZXhwaXJhdGlvbiI6ICItYMDA3LTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjoyWwogICAgeyJidWNmZDQ0IiwiaWF0eSI6MTUxNjc1OTk5LCJpcmlzaW9udCI6ImF1dG8ifQ==
--9431149156168

Content-Disposition: form-data; name="Signature"

ØRavWzkygo6QX9caELqKi9kDbU=
--9431149156168

Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
Content-Type: image/jpeg

...file content...
--9431149156168

Content-Disposition: form-data; name="submit"


Upload to Amazon S3
--9431149156168--
```

Risposta di esempio

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
Location: https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html?bucket=awsexamplebucket1&key=user/eric/MyPicture.jpg&etag="39d459dfbc0faabbb5e179358dfb94c3";
Server: AmazonS3
```

Caricamento di un'area di testo

Argomenti

- Costruzione di una policy e di un modulo
- Richiesta di esempio
- Risposta di esempio

L'esempio seguente illustra il processo completo di costruzione di una policy e di un modulo per caricare un'area di testo. Il caricamento di un'area di testo è utile per inviare contenuti creati dall'utente, come i post dei blog.

Costruzione di una policy e di un modulo

La policy seguente supporta caricamenti di aree di testo in Amazon S3 per il bucket `awsexamplebucket1`.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html"},
    ["eq", "$Content-Type", "text/html"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

Questa policy richiede quanto segue:

- Il caricamento deve avvenire prima delle 12:00, orario GMT, del 01/12/2007.
- Il contenuto deve essere caricato nel bucket `awsexamplebucket1`.
- La chiave deve iniziare con `"user/eric/»`.
- L'ACL è impostata su `public-read`.
- Il `success_action_redirect` è impostato su `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html`.
- L'oggetto è un testo HTML.
- Il tag `x-amz-meta-uuid` deve essere impostato su `14365123651274`.
- Il campo `x-amz-meta-tag` può contenere qualsiasi valore.

Di seguito è riportata una versione di questa politica. Base64-encoded

```
eyJhZlZxhwaXJhdGlvbiI6IClYMDA3LTExLTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXR
```

```
pb25zIjogWwogICAgeyJidWNrZXQiOiAiam9obnNtaXR0In0sCiAgICBbInN0YXJ0cy13aXR0IiwgIiRrZXkiLCaidXNlci
LAogICAgeyJhY2wiOiAicHVibGljLXJlYWQifSwKICAgIHsic3VjY2Vzc19hY3Rpb25fcmVkaXJlY3QiOiAiaHR0cDovL2p
C5zMy5hbWw6b25hd3MuY29tL25ld19wb3N0Lmh0bWwifSwKICAgIFsiZXEiLCAiJENvbnRlbnQtVHlwZSI6ICJ0ZXh0L2h0
CAgIHsieC1hbXotbWV0YS11dWlkIjogIjE0MzY1MTIzNjUxMjc0In0sCiAgICBbInN0YXJ0cy13aXR0IiwgIiR4LWFtei1t
IsICIiXQogIF0KfQo=
```

Creare una firma utilizzando le credenziali personali. Ad esempio, qA7FWXKq6VvU68lI9KdveT1cWgF= è la firma per il documento di policy precedente.

Il modulo seguente supporta una richiesta POST al amzn-s3-demo-bucket bucket che utilizza questa politica.

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
    ...
    <form action="https://amzn-s3-demo-bucket.s3.us-west-1.amazonaws.com/" method="post"
    enctype="multipart/form-data">
      Key to upload: <input type="input" name="key" value="user/eric/" /><br />
      <input type="hidden" name="acl" value="public-read" />
      <input type="hidden" name="success_action_redirect" value="https://
awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html" />
      <input type="hidden" name="Content-Type" value="text/html" />
      <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
      Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
      <input type="hidden" name="AWSAccessKeyId" value="AKIAIOSFODNN7EXAMPLE" />
      <input type="hidden" name="Policy" value="POLICY" />
      <input type="hidden" name="Signature" value="SIGNATURE" />
      Entry: <textarea name="file" cols="60" rows="10">
```

Your blog post goes here.

```
    </textarea><br />
    <!-- The elements after this will be ignored -->
    <input type="submit" name="submit" value="Upload to Amazon S3" />
  </form>
  ...
</html>
```

Richiesta di esempio

Questa richiesta presuppone che l'immagine caricata abbia una dimensione di 117.108 byte; i dati dell'immagine non sono inclusi.

```
POST / HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10) Gecko/20071115
  Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=178521717625888
Content-Length: 118635

-178521717625888
Content-Disposition: form-data; name="key"

ser/eric/NewEntry.html
--178521717625888
Content-Disposition: form-data; name="acl"

public-read
--178521717625888
Content-Disposition: form-data; name="success_action_redirect"

https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html
--178521717625888
Content-Disposition: form-data; name="Content-Type"

text/html
--178521717625888
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
--178521717625888
Content-Disposition: form-data; name="x-amz-meta-tag"

Interesting Post
--178521717625888
```

```
Content-Disposition: form-data; name="AWSAccessKeyId"

AKIAIOSFODNN7EXAMPLE
--178521717625888
Content-Disposition: form-data; name="Policy"
eyJhZlZxhwaXJhdGlvbiI6ICIyMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidW
--178521717625888
Content-Disposition: form-data; name="Signature"

qA7FWXKq6VvU68lI9KdveT1cWgF=
--178521717625888
Content-Disposition: form-data; name="file"

...content goes here...
--178521717625888
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--178521717625888--
```

Risposta di esempio

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
Location: https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html?
bucket=awsexamplebucket1&key=user/eric/
NewEntry.html&etag=40c3271af26b7f1672e41b8a274d28d4
Server: AmazonS3
```

POST con adobe flash (AWS firma (versione 2))

Questa sezione descrive come utilizzare POST con Adobe Flash.

Sicurezza di Adobe Flash Player

Per impostazione predefinita, il modello di sicurezza di Adobe Flash Player proibisce ad Adobe Flash Player di eseguire connessioni di rete con server al di fuori del dominio che serve il file SWF.

Per sostituire l'impostazione predefinita, è necessario caricare un file `crossdomain.xml` pubblicamente leggibile nel bucket che accetterà i caricamenti POST. Di seguito è riportato un esempio di file `crossdomain.xml`.

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
<allow-access-from domain="*" secure="false" />
</cross-domain-policy>
```

Note

Per ulteriori informazioni sul modello di sicurezza di Adobe Flash, visitare il sito Web di Adobe.

L'aggiunta del file `crossdomain.xml` al bucket permette la connessione di qualsiasi strumento Adobe Flash Player al file `crossdomain.xml` all'interno del bucket, ma non concede l'accesso all'effettivo bucket Amazon S3.

Considerazioni su Adobe Flash

L' `FileReference` API in Adobe Flash aggiunge il campo del `Filename` modulo alla richiesta POST. Quando crei applicazioni Adobe Flash che vengono caricate su Amazon S3 utilizzando l'azione `FileReference` API, includi la seguente condizione nella tua policy:

```
['starts-with', '$Filename', '']
```

Alcune versioni di Adobe Flash Player non gestiscono correttamente le risposte HTTP che hanno un corpo vuoto. Per configurare POST in modo che restituisca una risposta senza corpo vuoto, impostare `success_action_status` su 201. Amazon S3 restituisce quindi un documento XML con codice di stato 201. Per informazioni sul contenuto del documento XML, consulta [POST Object](#). Per informazioni sui campi del modulo, consulta [Campi del modulo HTML](#).

Appendice: API di configurazione del ciclo di vita (obsolete)

La configurazione del ciclo di vita del bucket viene aggiornata per supportare i filtri basati sui tag degli oggetti. Ovvero, ora puoi specificare una regola che specifichi il prefisso del nome della chiave, uno o

più tag oggetto o entrambi per selezionare un sottoinsieme di oggetti a cui si applica la regola. Le API sono state aggiornate di conseguenza. I seguenti argomenti descrivono la versione precedente delle operazioni del ciclo di vita dei bucket PUT e GET per la compatibilità con le versioni precedenti.

Argomenti

- [Ciclo di vita del bucket PUT \(obsoleto\)](#)
- [Ciclo di vita del bucket GET \(obsoleto\)](#)

Ciclo di vita del bucket PUT (obsoleto)

Description

Important

Per una versione aggiornata di questa API, consulta. [PutBucketLifecycleConfiguration](#)
Questa versione è stata dichiarata obsoleta. Le configurazioni esistenti del ciclo di vita funzioneranno. Per le nuove configurazioni del ciclo di vita, utilizza l'API aggiornata.

Crea una nuova configurazione del ciclo di vita per il bucket o sostituisce la configurazione del ciclo di vita esistente. Per informazioni sulla configurazione del ciclo di vita, consulta [Object Lifecycle Management](#) nella Amazon Simple Storage Service User Guide.

Permissions

Per impostazione predefinita, tutte le risorse Amazon S3, inclusi bucket, oggetti e relative risorse secondarie (ad esempio, la configurazione del ciclo di vita e la configurazione del sito Web) sono private. Solo il proprietario della risorsa, colui Account AWS che ha creato la risorsa, può accedervi. Il proprietario della risorsa può facoltativamente concedere autorizzazioni di accesso ad altri scrivendo una politica di accesso. Per questa operazione, gli utenti devono ottenere l'`s3:PutLifecycleConfiguration` autorizzazione.

Puoi anche negare esplicitamente le autorizzazioni. La negazione esplicita sostituisce anche qualsiasi altra autorizzazione. Se desideri impedire agli utenti o agli account di rimuovere o eliminare oggetti dal tuo bucket, devi negare loro le autorizzazioni per le seguenti azioni:

- `s3:DeleteObject`
- `s3:DeleteObjectVersion`
- `s3:PutLifecycleConfiguration`

Per ulteriori informazioni sulle autorizzazioni, consulta [Managing Access Permissions to Your Amazon S3 Resources](#) nella Amazon Simple Storage Service User Guide.

Richieste

Sintassi

```
PUT /?lifecycle HTTP/1.1
Host: bucketname.s3.amazonaws.com
Content-Length: length
Date: date
Authorization: authorization string
Content-MD5: MD5
```

Lifecycle configuration in the request body

Per informazioni dettagliate sulle stringhe di autorizzazione, consulta. [Autenticazione delle richieste \(AWS Signature \(versione 4\)\)](#)

Parametri della richiesta

Questa implementazione dell'operazione non utilizza parametri di richiesta.

Intestazioni di richiesta

Nome	Description	Richiesto
Content-MD5	Il digest MD5 a 128 bit con codifica base64 dei dati. È necessario utilizzare questa intestazione come controllo dell'integrità del messaggio per verificare che il corpo della richiesta non sia stato danneggiato durante il transito. Per ulteriori informazioni, vedere RFC 1864. Tipo: stringa Impostazione predefinita: nessuna	Sì

Corpo della richiesta

Nella richiesta, si specifica la configurazione del ciclo di vita nel corpo della richiesta. La configurazione del ciclo di vita è specificata come XML. Di seguito è riportato un esempio di configurazione del ciclo di vita di base. Specifica una regola. La regola `Prefix in the` identifica gli

oggetti a cui si applica la regola. La regola specifica anche due azioni (TransitionExpiration). Ogni azione specifica una tempistica in cui Amazon S3 deve eseguire l'azione. StatusIndica se la regola è abilitata o disattivata.

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>key-prefix</Prefix>
    <Status>rule-status</Status>
    <Transition>
      <Date>value</Date>
      <StorageClass>storage class</StorageClass>
    </Transition>
    <Expiration>
      <Days>value</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

Se lo stato del bucket è abilitato o sospeso, puoi avere più versioni dello stesso oggetto: una versione corrente e zero o più versioni non correnti. La seguente configurazione del ciclo di vita specifica le azioni (,) specifiche per le versioni degli oggetti non correnti. NoncurrentVersionTransition NoncurrentVersionExpiration

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>key-prefix</Prefix>
    <Status>rule-status</Status>
    <NoncurrentVersionTransition>
      <NoncurrentDays>value</NoncurrentDays>
      <StorageClass>storage class</StorageClass>
    </NoncurrentVersionTransition>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>value</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
</LifecycleConfiguration>
```

L'API per il caricamento in più parti consente di caricare oggetti di grandi dimensioni in parti. Per ulteriori informazioni sui caricamenti in più parti, consulta la panoramica sui caricamenti in più

[parti](#) nella Amazon Simple Storage Service User Guide. Con la configurazione del ciclo di vita, puoi dire ad Amazon S3 di annullare i caricamenti multiparte incompleti, identificati dal prefisso del nome chiave specificato nella regola, se non vengono completati entro un determinato numero di giorni. Quando Amazon S3 annulla un caricamento in più parti, elimina tutte le parti associate al caricamento. Ciò garantisce che non si verifichino caricamenti multiparte incompleti che hanno lasciato parti archiviate in Amazon S3, quindi non è necessario pagare costi di archiviazione. Di seguito è riportata una configurazione del ciclo di vita di esempio che specifica una regola con l'operazione `AbortIncompleteMultipartUpload`. Questa azione indica ad Amazon S3 di annullare i caricamenti multiparte incompleti sette giorni dopo l'avvio.

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>SomeKeyPrefix</Prefix>
    <Status>rule-status</Status>
    <AbortIncompleteMultipartUpload>
      <DaysAfterInitiation>7</DaysAfterInitiation>
    </AbortIncompleteMultipartUpload>
  </Rule>
</LifecycleConfiguration>
```

La tabella seguente descrive gli elementi XML nella configurazione del ciclo di vita.

Nome	Description	Richiesto
AbortIncompleteMultipartUpload	Contenitore per specificare quando un caricamento multiparte incompleto diventa idoneo per un'operazione di interruzione. Figlio: DaysAfterInitiation Tipo: Container Antenato: Rule	Sì, se non viene specificata nessun'altra azione per la regola
Date	Data in cui desideri che Amazon S3 intraprenda l'azione. Per ulteriori informazioni, consulta Lifecycle Rules: Based on a Specific Date	Sì, se Days e Expired0b

Nome	Description	Richiesto
	nella Amazon Simple Storage Service User Guide. Il valore della data deve essere conforme al formato ISO 8601. L'ora è sempre la mezzanotte UTC. Tipo: stringa Antenato: o <code>Expiration Transition</code>	<code>jectDeleteMarker</code> sono assenti
Days	Specifica il numero di giorni dopo la creazione dell'oggetto in cui l'azione specifica della regola ha effetto. Tipo: numero intero non negativo se usato con <code>Transition</code> , numero intero positivo se usato con <code>Expiration</code> Antenato: <code>Expiration Transition</code>	Sì, se <code>Date</code> e <code>ExpiredObjectDeleteMarker</code> sono assenti
DaysAfterInitiation	Specifica il numero di giorni dopo l'avvio di un caricamento in più parti in cui il caricamento in più parti deve essere completato. Se non viene completato entro il numero di giorni specificato, diventa idoneo per un'operazione di interruzione e Amazon S3 annulla il caricamento multiparte incompleto. Tipo: numero intero positivo Antenato: <code>AbortIncompleteMultipartUpload</code>	Sì, se viene specificato un tag principale

Nome	Description	Richiesto
Expiration	Questa azione specifica un periodo della durata di un oggetto in cui Amazon S3 deve intraprendere l'azione di scadenza appropriata. L'azione intrapresa da Amazon S3 dipende dal fatto che il bucket sia abilitato al controllo delle versioni. • Se il controllo delle versioni non è mai stato abilitato nel bucket, Amazon S3 elimina definitivamente l'unica copia dell'oggetto. • Se il bucket è abilitato al controllo delle versioni (o il controllo delle versioni è sospeso), l'azione si applica solo alla versione corrente dell'oggetto. Un bucket abilitato al controllo delle versioni può avere molte versioni dello stesso oggetto: una versione corrente e zero o più versioni non correnti. Invece di eliminare la versione corrente, Amazon S3 la rende una versione non corrente aggiungendo un indicatore di eliminazione come nuova versione corrente.	Sì, se nessun'altra azione è presente nelRule.

 **Important**

Se lo stato di un bucket è sospeso dalla versione, Amazon S3 crea un indicatore di eliminazione con l'ID della versione. null Se disponi di una versione con ID di

versionenu11, Amazon S3 sovrascrive tale versione.

 **Note**

Per impostare la scadenza per gli oggetti non correnti, usa l'azione `NoncurrentVersionExpiration`

Nome	Description	Richiesto
LifecycleConfiguration	Container per le regole del ciclo di vita. Puoi aggiungere fino a 1.000 regole. Tipo: Container Bambini: Regola Antenato: Nessuno	Sì
ExpiredObjectDeleteMarker	In un bucket con versione (un bucket con versione abilitata o con versione sospesa), puoi aggiungere questo elemento nella configurazione del ciclo di vita per dire ad Amazon S3 di eliminare gli indicatori di eliminazione degli oggetti scaduti. Per un esempio, consulta Example 8: Removing Expired Object Delete Markers nella Amazon Simple Storage Service User Guide. Non aggiungerlo a un bucket senza versione, perché quel tipo di bucket non può includere indicatori di eliminazione. Tipo: stringa Valori validi: true false (il valore false è consentito, ma non è operativo, il che significa che Amazon S3 non interverrà) Antenato: Expiration	Sì, se Date e Days sono assenti

Nome	Description	Richiesto
NoncurrentDays	Specifica il numero di giorni durante i quali un oggetto è un oggetto noncurrent prima che Amazon S3 possa eseguire l'operazione associata. Per informazioni sui calcoli dei giorni non correnti, consulta How Amazon S3 calcola quando un oggetto è diventato non corrente nella Amazon Simple Storage Service User Guide. Tipo: numero intero non negativo se usato con, numero intero positivo se usato con NoncurrentVersionTransition NoncurrentVersionExpiration Antenato: o NoncurrentVersionExpiration NoncurrentVersionTransition	Sì
NoncurrentVersionExpiration	Specifica quando scadono le versioni degli oggetti non correnti. Alla scadenza, Amazon S3 elimina in modo definitivo le versioni degli oggetti non correnti. Imposta questa azione di configurazione del ciclo di vita su un bucket con il controllo delle versioni abilitato (o sospeso) per dire ad Amazon S3 di eliminare le versioni non correnti degli oggetti in un periodo specifico della durata dell'oggetto. Tipo: Container Bambini: NoncurrentDays Antenato: Regola	Sì, se nessun'altra azione è presente nel Rule

Nome	Description	Richiesto
NoncurrentVersionTransition	Contenitore per la regola di transizione che descrive quando gli oggetti non correnti passano alla classe di archiviazione STANDARD_IA ONEZONE_IA , o. Se il bucket è abilitato al controllo delle versioni (o se il controllo delle versioni è sospeso), puoi impostare questa azione per dire ad Amazon S3 di effettuare la transizione delle versioni non correnti degli oggetti in un periodo specifico della durata dell'oggetto. Tipo: Container Bambini: e NoncurrentDays StorageClass Antenato: Regola	Sì, se nessun'altra azione è presente nel Rule
Prefix	Prefisso chiave dell'oggetto che identifica uno o più oggetti a cui si applica la regola. Tipo: stringa Antenato: regola	Sì
Rule	Contenitore per una regola del ciclo di vita. Una configurazione del ciclo di vita può contenere fino a 1000 regole. Tipo: Container Ancestor:LifecycleConfiguration	Sì

Nome	Description	Richiesto
Status	Se abilitata, Amazon S3 esegue la regola come pianificato. Se è disattivata, Amazon S3 ignora la regola. Tipo: stringa Antenato: regola Valori validi: Abilitato, Disabilitato	Sì
StorageClass	Specifica la classe di storage Amazon S3 verso la quale desideri effettuare la transizione dell'oggetto. Tipo: stringa Antenato: transizione e NoncurrentVersionTransition Valori validi: STANDARD_IA ONEZONE_IA GLACIER	Sì Questo elemento è obbligatorio solo se si specifica uno o entrambi i suoi antenati.

Nome	Description	Richiesto
Transition	Questa azione specifica un periodo della durata degli oggetti in cui Amazon S3 deve effettuare la transizione alla <code>STANDARD_IA</code>, <code>ONEZONE_IA</code>, o classe di storage. Quando questa azione è attiva, ciò che fa Amazon S3 dipende dal fatto che il bucket sia abilitato al controllo delle versioni. • Se il controllo delle versioni non è mai stato abilitato nel bucket, Amazon S3 trasferisce l'unica copia dell'oggetto nella classe di storage specificata. • Se il bucket è abilitato al versioning (o il controllo delle versioni è sospeso), Amazon S3 esegue la transizione solo delle versioni correnti degli oggetti identificati nella regola.  Note Un bucket abilitato al controllo delle versioni può avere molte versioni di un oggetto. Questa azione non ha effetto sulle versioni degli oggetti non correnti. Per effettuare la transizione di oggetti non correnti, è necessario utilizzare l'azione <code>NoncurrentVersionTransition</code>. Tipo: Container	Sì, se nessun'altra azione è presente nel <code>Rule</code>

Nome	Description	Richiesto
	Bambini: giorni o data e StorageClass	
	Antenato: Regola	

Risposte

Intestazioni di risposta

Questa implementazione dell'operazione utilizza solo le intestazioni di risposta comuni alla maggior parte delle risposte. Per ulteriori informazioni, vedere [Common Response Headers](#).

Elementi di risposta

Questa implementazione dell'operazione non restituisce elementi di risposta.

Errori speciali

Questa implementazione dell'operazione non restituisce errori speciali. Per informazioni generali sugli errori di Amazon S3 e un elenco di codici di errore, consulta [Error Responses](#).

Esempi

Esempio 1: aggiungere la configurazione del ciclo di vita a un bucket che non lo è Versioning-enabled

La seguente configurazione del ciclo di vita specifica due regole, ciascuna con un'azione.

- L'azione Transition indica ad Amazon S3 di trasferire gli oggetti con il prefisso «documents/» alla classe di storage 30 giorni dopo la creazione.
- L'azione Expiration indica ad Amazon S3 di eliminare gli oggetti con il prefisso «logs/» 365 giorni dopo la creazione.

```
<LifecycleConfiguration>
  <Rule>
    <ID>id1</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <Transition>
```

```

    <Days>30</Days>
    <StorageClass>GLACIER</StorageClass>
  </Transition>
</Rule>
<Rule>
  <ID>id2</ID>
  <Prefix>logs</Prefix>
  <Status>Enabled</Status>
  <Expiration>
    <Days>365</Days>
  </Expiration>
</Rule>
</LifecycleConfiguration>

```

Di seguito è riportato un esempio di PUT `/?lifecycle` richiesta che aggiunge la precedente configurazione del ciclo di vita al bucket. `examplebucket`

```

PUT /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Wed, 14 May 2014 02:11:21 GMT
Content-MD5: q6yJD1IkBaGGfb3QLY69A==
Authorization: authorization string
Content-Length: 415

<LifecycleConfiguration>
  <Rule>
    <ID>id1</ID>
    <Prefix>documents</Prefix>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
  <Rule>
    <ID>id2</ID>
    <Prefix>logs</Prefix>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>

```


Di seguito è riportata una risposta di esempio.

```
HTTP/1.1 200 OK
x-amz-id-2: r+qR7+nhXtJDDIJ0JJYcd+1j5nM/rUFiiiZ/fNbD0sd3JUE8NWMLNHXmvPfwMpdC
x-amz-request-id: 9E26D08072A8EF9E
Date: Wed, 14 May 2014 02:11:22 GMT
Content-Length: 0
Server: AmazonS3
```

Esempio 2: aggiungere la configurazione del ciclo di vita a un bucket Versioning-enabled

La seguente configurazione del ciclo di vita specifica due regole, ciascuna con un'azione da eseguire per Amazon S3. Specifichi queste azioni quando il bucket è abilitato al versioning o il controllo delle versioni è sospeso:

- L'NoncurrentVersionExpirationazione indica ad Amazon S3 di far scadere le versioni non correnti degli oggetti con il prefisso «logs/» 100 giorni dopo che gli oggetti sono diventati non correnti.
- L'NoncurrentVersionTransitionazione indica ad Amazon S3 di trasferire le versioni non correnti degli oggetti con il prefisso «documents/» alla classe di storage 30 giorni dopo che sono diventate non correnti.

```
<LifecycleConfiguration>
  <Rule>
    <ID>DeleteAfterBecomingNonCurrent</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>100</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
  <Rule>
    <ID>TransitionAfterBecomingNonCurrent</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionTransition>
      <NoncurrentDays>30</NoncurrentDays>
      <StorageClass>GLACIER</StorageClass>
    </NoncurrentVersionTransition>
  </Rule>
```

```
</LifecycleConfiguration>
```

Di seguito è riportato un esempio di PUT `/?lifecycle` richiesta che aggiunge la precedente configurazione del ciclo di vita al bucket. `examplebucket`

```
PUT /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Wed, 14 May 2014 02:21:48 GMT
Content-MD5: 96rxH9mDqVnKkaZDddgnw==
Authorization: authorization string
Content-Length: 598

<LifecycleConfiguration>
  <Rule>
    <ID>DeleteAfterBecomingNonCurrent</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>1</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
  <Rule>
    <ID>TransitionSoonAfterBecomingNonCurrent</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionTransition>
      <NoncurrentDays>0</NoncurrentDays>
      <StorageClass>GLACIER</StorageClass>
    </NoncurrentVersionTransition>
  </Rule>
</LifecycleConfiguration>
```

Di seguito è riportata una risposta di esempio.

```
HTTP/1.1 200 OK
x-amz-id-2: aXQ+KbIrmMmo0//3bMdDTw/CnjArwje+J49Hf+j44yRb/VmbIkgI05A+PT98Cp/6k07hf+LD2mY=
x-amz-request-id: 02D7EC4C10381EB1
Date: Wed, 14 May 2014 02:21:50 GMT
Content-Length: 0
Server: AmazonS3
```

Esempi aggiuntivi

Per altri esempi di transizione di oggetti verso classi di archiviazione come STANDARD_IA o ONEZONE_IA, vedi Esempi di configurazione del ciclo di vita. <https://docs.aws.amazon.com/AmazonS3/latest/userguide/intro-lifecycle-rules.html#lifecycle-configuration-examples>

Risorse correlate

- [GetBucketLifecycleConfiguration](#)
- [POST Object restore](#)
- Per impostazione predefinita, un proprietario della risorsa, in questo caso, il proprietario del bucket, ovvero colui che ha creato il bucket, può eseguire qualsiasi operazione. Account AWS Il proprietario di una risorsa può anche concedere ad altri l'autorizzazione per eseguire l'operazione. Per ulteriori informazioni, consulta i seguenti argomenti nella Guida per l'utente di Amazon Simple Storage Service:
 - [Specificare le autorizzazioni in una policy](#)
 - [Gestione delle autorizzazioni di accesso alle risorse di Amazon S3](#)

Ciclo di vita del bucket GET (obsoleto)

Description

Important

Per una versione aggiornata di questa API, consulta [GetBucketLifecycleConfiguration](#). Se hai configurato un ciclo di vita del bucket utilizzando l'<filter>elemento, dovresti vedere una versione aggiornata di questo argomento. Questo argomento viene fornito per motivi di compatibilità con le versioni precedenti.

Restituisce le informazioni `lifecycle` di configurazione impostate nel bucket. Per informazioni sulla configurazione del ciclo di vita, consulta [Object Lifecycle Management](#) nella Amazon Simple Storage Service User Guide.

Per utilizzare questa operazione, è necessario disporre dell'autorizzazione per eseguire l'azione. `s3:GetLifecycleConfiguration` Il proprietario del bucket dispone di questa autorizzazione per impostazione predefinita. Il proprietario del bucket può concedere questa autorizzazione ad altri. Per ulteriori informazioni sulle autorizzazioni, consulta [Managing Access Permissions to Your Amazon S3 Resources](#) nella Amazon Simple Storage Service User Guide.

Richieste

Sintassi

```
GET /?lifecycle HTTP/1.1
Host: bucketname.s3.amazonaws.com
Date: date
Authorization: authorization string (see Autenticazione delle richieste \(AWS Signature \(versione 4\)\))
```

Parametri della richiesta

Questa implementazione dell'operazione non utilizza parametri di richiesta.

Intestazioni di richiesta

Questa implementazione dell'operazione utilizza solo intestazioni di richiesta comuni a tutte le operazioni. Per ulteriori informazioni, vedere [Common Request Headers](#).

Elementi della richiesta

Questa implementazione dell'operazione non utilizza elementi di richiesta.

Risposte

Intestazioni di risposta

Questa implementazione dell'operazione utilizza solo le intestazioni di risposta comuni alla maggior parte delle risposte. Per ulteriori informazioni, vedere [Common Response Headers](#).

Elementi di risposta

Questa implementazione di GET restituisce i seguenti elementi di risposta.

Nome	Description	Richiesto
<code>AbortIncompleteMultipartUpload</code>	Contenitore per specificare quando un caricamento in più parti incompleto diventa idoneo per un'operazione di interruzione. Figlio: <code>DaysAfterInitiation</code> Tipo: Container Antenato: Rule	Sì, se non viene specificata nessun'altra azione per la regola
<code>Date</code>	Data in cui desideri che Amazon S3 intraprenda l'azione. Per ulteriori informazioni, consulta Lifecycle Rules: Based on a Specific Date nella Amazon Simple Storage Service User Guide. Il valore della data deve essere conforme allo standard ISO 8601. L'ora è sempre la mezzanotte UTC. Tipo: stringa Ancestor: <code>Expiration</code> <code>Transition</code>	Sì, se <code>Days</code> e <code>ExpiredObjectDeleteMarker</code> sono assenti

Nome	Description	Richiesto
Days	Specifica il numero di giorni dopo la creazione dell'oggetto in cui l'azione specifica della regola ha effetto. Il tempo di idoneità dell'oggetto viene calcolato come l'ora di creazione + il numero di giorni con l'ora risultante arrotondata alla mezzanotte UTC del giorno successivo. Tipo: Non-negative intero se usato con <code>Transition</code> , numero intero positivo se usato con <code>Expiration</code> Antenato: <code>o Transition Expiration</code>	Sì, se <code>Date</code> e <code>ExpiredObjectDeleteMarker</code> sono assenti
DaysAfterInitiation	Specifica il numero di giorni dopo l'avvio di un caricamento in più parti in cui il caricamento in più parti deve essere completato. Se non viene completato entro il numero di giorni specificato, diventa idoneo per un'operazione di interruzione e Amazon S3 annulla il caricamento multiparte incompleto. Tipo: numero intero positivo Antenato: <code>AbortIncompleteMultipartUpload</code>	Sì, se <code>Date</code> è assente

Nome	Description	Richiesto
Expiration	Questa azione specifica un periodo della durata dell'oggetto in cui Amazon S3 deve intraprendere l'azione di scadenza appropriata. L'azione di scadenza si verifica solo sugli oggetti idonei in base al periodo specificato nel figlio Date o Days nell'elemento. L'azione intrapresa da Amazon S3 dipende dal fatto che il bucket sia abilitato per il controllo delle versioni. • Se il controllo delle versioni non è mai stato abilitato nel bucket, Amazon S3 elimina definitivamente l'unica copia dell'oggetto. • Altrimenti, se il bucket è abilitato al controllo delle versioni (o il controllo delle versioni è sospeso), l'azione si applica solo alla versione corrente dell'oggetto. I bucket abilitati o sospesi per il controllo delle versioni possono avere molte versioni dello stesso oggetto: una versione corrente e zero o più versioni non correnti. Invece di eliminare la versione corrente, Amazon S3 la rende una versione non corrente aggiungendo un indicatore di eliminazione come nuova versione corrente.  Important Se lo stato di un bucket è sospeso dalla versione, Amazon S3 crea un indicatore di eliminazione	Sì, se viene specificato il tag principale

Nome	Description	Richiesto
	con l'ID della versione. null Se disponi di una versione con ID di versionenull, Amazon S3 sovrascrive tale versione. Note Per impostare la scadenza per gli oggetti non correnti, è necessario utilizzare l'azione. NoncurrentVersionExpiration Tipo: Container Bambini: giorni o data Antenato: Regola	
ID	Identificatore univoco per la regola. Il valore non può essere più lungo di 255 caratteri. Tipo: stringa Antenato: regola	No

Nome	Description	Richiesto
LifecycleConfiguration	Container per le regole del ciclo di vita. Puoi aggiungere fino a 1.000 regole. Tipo: Container Bambini: Regola Antenato: Nessuno	Sì
ExpiredObjectDeleteMarker	In un bucket con versione (con versione abilitata o con versione sospesa), questo elemento indica se Amazon S3 eliminerà eventuali indicatori di eliminazione di oggetti scaduti nel bucket. Per un esempio, vai all'Esempio 8: Specifica l'azione di scadenza per rimuovere gli indicatori di eliminazione di oggetti scaduti nella Guida per l'utente di Amazon Simple Storage Service. Tipo: stringa Valori validi: true false (il valore false è consentito ma non è operativo, Amazon S3 non interviene se il valore è falso) Antenato: Expiration	Sì, se Date e Days sono assenti

Nome	Description	Richiesto
NoncurrentDays	Specifica il numero di giorni in cui un oggetto non è corrente prima che Amazon S3 possa eseguire l'azione associata. Per informazioni sul calcolo dei giorni non correnti, consulta le regole del ciclo di vita basate sul numero di giorni nella Amazon Simple Storage Service User Guide. Tipo: numero intero non negativo se usato con, numero intero positivo se usato con NoncurrentVersionTransition NoncurrentVersionExpiration Antenato: o NoncurrentVersionExpiration NoncurrentVersionTransition	Sì, solo se l'antenato è presente
NoncurrentVersionExpiration	Specifica quando scadono le versioni degli oggetti non correnti. Alla scadenza, Amazon S3 elimina in modo definitivo le versioni degli oggetti non correnti. Imposta questa azione di configurazione del ciclo di vita su un bucket con il controllo delle versioni abilitato (o sospeso) per richiedere ad Amazon S3 di eliminare le versioni non correnti degli oggetti in un periodo specifico della durata dell'oggetto. Tipo: Container Bambini: NoncurrentDays Antenato: Regola	Sì, se non è presente nessun'altra azione nel Rule

Nome	Description	Richiesto
NoncurrentVersionTransition	Contenitore per la regola di transizione che descrive quando gli oggetti non correnti passano alla STANDARD_IA , ONEZONE_IA , o alla classe di archiviazione. Se il bucket è abilitato al controllo delle versioni (o il controllo delle versioni è sospeso), puoi impostare questa azione per richiedere ad Amazon S3 di effettuare la transizione delle versioni non correnti degli oggetti alla classe di storage in un periodo specifico della durata dell'oggetto. Tipo: Container Bambini: e NoncurrentDays StorageClass Antenato: Regola	Sì, se non è presente nessun'altra azione nel Rule
Prefix	Prefisso chiave dell'oggetto che identifica uno o più oggetti a cui si applica la regola. Tipo: stringa Antenato: regola	Sì
Rule	Contenitore per una regola del ciclo di vita. Tipo: Container Antenato: LifecycleConfiguration	Sì

Nome	Description	Richiesto
Status	Se abilitata, Amazon S3 esegue la regola come pianificato. Se disabilitato, Amazon S3 ignora la regola. Tipo: stringa Antenato: regola Valori validi: abilitato o disabilitato	Sì
StorageClass	Specifica la classe di storage Amazon S3 a cui desideri effettuare la transizione dell'oggetto. Tipo: stringa Antenato: transizione e NoncurrentVersionTransition Valori validi: STANDARD_IA ONEZONE_IA	Sì

Nome	Description	Richiesto
Transition	Questa azione specifica un periodo della durata degli oggetti in cui Amazon S3 deve effettuare la transizione alla <code>STANDARD_IA</code>, <code>ONEZONE_IA</code>, o classe di storage. Quando questa azione è attiva, ciò che fa Amazon S3 dipende dal fatto che il bucket sia abilitato al controllo delle versioni. • Se il controllo delle versioni non è mai stato abilitato nel bucket, Amazon S3 trasferisce l'unica copia dell'oggetto nella classe di storage specificata. • Quando il bucket è abilitato al versioning (o il controllo delle versioni è sospeso), Amazon S3 esegue la transizione solo delle versioni correnti degli oggetti identificati nella regola.  Note Un bucket con versione abilitata o sospesa può contenere molte versioni di un oggetto. Questa azione non ha effetto sulle versioni non correnti dell'oggetto. Per eseguire la transizione di oggetti non correnti, è necessario utilizzare l'azione <code>NoncurrentVersionTransition</code>.	Sì, se non è presente nessun'altra azione nel Rule

Nome	Description	Richiesto
	Tipo: Container Bambini: giorni o data e StorageClass Antenato: Regola	

Errori speciali

Codice di errore	Description	Codice di stato HTTP	Prefisso del codice di errore SOAP
NoSuchLifecycleConfiguration	La configurazione del ciclo di vita non esiste.	404 Not Found (404 Non trovato)	Client

Per informazioni generali sugli errori di Amazon S3 e un elenco di codici di errore, consulta. [Risposte agli errori](#)

Esempi

Esempio 1: recupero di una sottorisorsa del ciclo di vita

Questo esempio è una richiesta GET per recuperare la `lifecycle` sottorisorsa dal bucket specificato e una risposta di esempio con la configurazione del ciclo di vita restituita.

Richiesta di esempio

```
GET /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Thu, 15 Nov 2012 00:17:21 GMT
Authorization: signatureValue
```

Risposta di esempio

```
HTTP/1.1 200 OK
```

```
x-amz-id-2: ITnGT1y4RyTmXa3rPi4hk1TXouTf0hccUjo0iCPjz6FnfIutBj3M7fPG1W02SEWp
x-amz-request-id: 51991C342C575321
Date: Thu, 15 Nov 2012 00:17:23 GMT
Server: AmazonS3
Content-Length: 358
```

```
<?xml version="1.0" encoding="UTF-8"?>
<LifecycleConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Rule>
    <ID>Archive and then delete rule</ID>
    <Prefix>projectdocs</Prefix>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>STANDARD_IA</StorageClass>
    </Transition>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>3650</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

Risorse correlate

- [PutBucketLifecycleConfiguration](#)
- [DeleteBucketLifecycle](#)

Cronologia dei documenti

La tabella seguente descrive le modifiche importanti alla Amazon S3 Developer Guide. Per ricevere notifiche sugli aggiornamenti di questa documentazione, è possibile iscriversi a un feed RSS.

Modifica	Descrizione	Data
Nuova guida per sviluppatori	Tutti i contenuti non API, come lo sviluppo con S3, gli esempi di codice, le risposte agli errori e altro ancora, sono stati spostati dalla guida di riferimento alle API di Amazon S3 in questa nuova guida per sviluppatori di Amazon S3. Per ulteriori informazioni, consulta Amazon S3 API Reference .	15 giugno 2026

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.