



Panduan Developer

Amazon Simple Storage Service



Versi API 2006-03-01

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

Amazon Simple Storage Service: Panduan Developer

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

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan antara para pelanggan, atau dengan cara apa pun yang menghina atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan milik dari pemiliknya masing-masing, yang mungkin berafiliasi dengan, terhubung ke, atau disponsori oleh Amazon.

Table of Contents

Apa itu Amazon S3?	1
Rekomendasi panggilan API	2
Membuat panggilan REST API langsung	2
Perizinan	3
Mengembangkan dengan Amazon S3	4
Membuat permintaan	4
Tentang kunci akses	5
Titik akhir permintaan	7
Membuat permintaan melalui IPv6	7
Membuat permintaan menggunakan AWS SDK	18
Membuat permintaan menggunakan API REST	57
Menggunakan AWS CLI	70
Pelajari lebih lanjut tentang AWS CLI	71
Mengembangkan dengan AWS SDK	71
Antarmuka Pemrograman SDK	72
Menentukan Versi Tanda Tangan dalam otentikasi permintaan	73
Dapatkan ID permintaan Amazon S3 untuk Dukungan	82
Menggunakan AWS CLI untuk mendapatkan ID permintaan	83
Menggunakan Windows PowerShell untuk mendapatkan ID permintaan	84
Penggunaan AWS CloudTrail peristiwa data untuk mendapatkan ID permintaan	84
Menggunakan pencatatan log akses server S3 untuk mendapatkan ID permintaan	84
Menggunakan HTTP untuk memperoleh ID Permintaan	84
Menggunakan peramban web untuk memperoleh ID permintaan	85
Menggunakan AWS SDK untuk mendapatkan ID permintaan	85
Operasi API tingkat objek S3 yang didukung untuk Tabel S3	90
Contoh kode	96
Amazon S3	102
Hal-hal mendasar	107
Skenario	799
Contoh nirserver	1254
Kontrol Amazon S3	1266
Hal-hal mendasar	1267
Ember Direktori S3	1346
Hal-hal mendasar	1348

Skenario	1473
Permintaan Otentikasi (AWS Versi Tanda Tangan 4)	1477
Metode Autentikasi	1478
Pengantar Permintaan Penandatanganan	1479
Menggunakan Header Otorisasi	1481
Gambaran umum	1481
Perhitungan Tanda Tangan: Transfer Payload dalam Satu Potongan	1486
Perhitungan Tanda Tangan: Transfer Payload dalam Beberapa Potongan	1506
Perhitungan Tanda Tangan: Termasuk Header Trailing	1520
Menggunakan Parameter Kueri	1526
Menghitung Tanda Tangan	1530
Sebuah contoh	1535
Contoh 2	1537
Contoh: Perhitungan Tanda Tangan	1537
Contoh Perhitungan Tanda Tangan Menggunakan Java	1538
Contoh Perhitungan Tanda Tangan Menggunakan C #	1539
Mengotentikasi Permintaan HTTP POST	1540
Menghitung Tanda Tangan	1543
Kunci Kebijakan Khusus Otentikasi Tanda Tangan Amazon S3 Versi 4	1544
Contoh kebijakan bucket menggunakan kunci kondisi terkait Signature Versi 4	1547
Browser-Based Unggah Menggunakan POST	1551
Objek POST	1552
Deskripsi	1552
Penentuan versi	1553
Permohonan	1553
Contoh	1573
Sumber Daya Terkait	1574
Pemulihan objek POST	1575
Deskripsi	1575
Mengkueri Arsip dengan Permintaan Pilihan	1575
Memulihkan Arsip	1577
Permintaan	1579
Respons	1594
Contoh	1595
Info Selengkapnya	1598
Browser-Based Unggah Menggunakan HTTP POST	1598

Menghitung Tanda Tangan	1600
Membuat Formulir HTML	1601
Deklarasi Formulir HTML	1602
Bidang Formulir HTML	1603
Kebijakan POST	1610
Kedaluwarsa	1611
Pencocokan Kondisi	1611
Ketentuan	1613
Karakter Melarikan Diri	1618
Contoh Upload POST	1619
Mengunggah File ke Amazon S3 Menggunakan HTTP POST	1619
Browser-based unggah menggunakan Amplify AWS	1622
Menggunakan AWS Memperkuat per JavaScript pustakaan untuk Mengunggah File ke Amazon S3	1622
Info Selengkapnya	1623
Header permintaan umum	1624
Header respons umum	1629
Tanggapan kesalahan	1634
Respons kesalahan REST	1635
Daftar kode kesalahan	1637
Daftar Kode Kesalahan Konten Objek SELECT	1663
Daftar kode Replication-related kesalahan	1676
Daftar kode Tagging-related kesalahan	1679
Daftar kode kesalahan terkait anotasi Amazon S3	1680
Daftar kode kesalahan Amazon S3 di Outposts	1684
Daftar kode kesalahan Amazon S3 Storage Lens	1685
Daftar kode kesalahan Lambda Objek Amazon S3	1694
Daftar kode kesalahan asinkron Amazon S3	1698
Daftar kode kesalahan Amazon S3 Access Grants	1700
Daftar kode FSx-related kesalahan Amazon	1703
Daftar kode kesalahan Amazon S3 Tables	1704
Praktik terbaik kesalahan Amazon S3	1705
Coba lagi InternalErrors	1705
Setel aplikasi untuk SlowDown kesalahan berulang	1705
Isolasi kesalahan	1706
AWS Glosarium	1707

Sumber daya	1708
Lampiran	1710
Lampiran: Tang SelectObjectContent gapan	1711
Deskripsi	1711
Respons	1711
Sumber Daya Terkait	1721
Lampiran: objek OPTIONS	1722
Deskripsi	1722
Permintaan	1722
Respons	1723
Contoh	1725
Sumber Daya Terkait	1725
Lampiran: SOAP API	1726
Operasi pada Layanan (SOAP API)	1726
Operasi pada Bucket (SOAP API)	1728
Operasi pada Objek (SOAP API)	1742
Mengautentikasi permintaan SOAP	1767
Mengatur kebijakan akses dengan SOAP	1768
Elemen-elemen umum	1770
Tanggapan Kesalahan SOAP	1770
Lampiran: Permintaan otentikasi (AWS tanda tangan versi 2)	1772
Mengotentikasi permintaan menggunakan REST API (AWS tanda tangan versi 2)	1774
Menandatangani dan mengotentikasi permintaan REST (AWS tanda tangan versi 2)	1777
Browser-based unggah menggunakan POST (AWS tanda tangan versi 2)	1792
Lampiran: API Konfigurasi Siklus Hidup (Tidak Digunakan)	1817
Siklus hidup PUT Bucket (Tidak digunakan lagi)	1819
Siklus hidup GET Bucket (Tidak digunakan lagi)	1836
Riwayat dokumen	1848
.....	mdcccclix

Apa itu Amazon S3?

Amazon Simple Storage Service (Amazon S3) adalah layanan penyimpanan objek yang menawarkan skalabilitas, ketersediaan data, keamanan, dan kinerja terdepan di industri. Pelanggan dari semua ukuran dan industri dapat menggunakan Amazon S3 untuk menyimpan dan melindungi sejumlah data untuk berbagai kasus penggunaan, seperti danau data, situs web, aplikasi seluler, pencadangan dan pemulihan, arsip, aplikasi perusahaan, perangkat IoT, dan analitik big data. Amazon S3 menyediakan fitur manajemen sehingga Anda dapat mengoptimalkan, mengatur, dan mengonfigurasi akses ke data Anda untuk memenuhi persyaratan bisnis, organisasi, dan kepatuhan spesifik Anda.

Panduan ini memberikan informasi yang berfokus pada pengembang tentang penggunaan API REST Amazon S3 untuk membangun aplikasi yang menyimpan dan mengambil data di cloud.

Anda dapat menggunakan toolkit yang mendukung HTTP untuk menggunakan API REST. Anda bahkan dapat menggunakan peramban untuk mengambil objek, selama objek tersebut dapat dibaca secara anonim.

API REST menggunakan header HTTP standar dan kode status, sehingga peramban dan toolkit standar bekerja sesuai harapan. Di beberapa area, kami menambahkan fungsionalitas ke HTTP (misalnya, kami menambahkan header untuk mendukung kontrol akses). Dalam kasus ini, kami menambahkan fungsionalitas baru dengan cara yang sesuai dengan gaya penggunaan HTTP standar.

Note

Dukungan untuk SOAP melalui HTTP tidak digunakan lagi, tetapi masih tersedia melalui HTTPS. Namun, fitur baru Amazon S3 tidak akan didukung untuk SOAP. Kami menyarankan Anda menggunakan REST API ini atau AWS SDK di tautan berikut:

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

Panduan pengembang ini meliputi:

- [Header permintaan umum](#) dan [Header respons umum](#)
- [Tanggapan kesalahan](#)
- [Browser-Based Unggah Menggunakan POST \(AWS Versi Tanda Tangan 4\)](#)

 Important

Baca berikut ini tentang otentikasi dan kontrol akses sebelum pergi ke topik API tertentu.

Permintaan ke Amazon S3 dapat diautentikasi atau anonim. Akses yang diautentikasi memerlukan kredentif yang AWS dapat digunakan untuk mengotentikasi permintaan Anda.

Rekomendasi panggilan API

Membuat panggilan REST API langsung dari kode Anda bisa merepotkan. Ini mengharuskan Anda untuk menulis kode yang diperlukan untuk menghitung tanda tangan yang valid untuk mengotentikasi permintaan Anda. Kami merekomendasikan alternatif berikut sebagai gantinya:

- Gunakan [AWS SDK](#) untuk mengirim permintaan Anda.

Juga, lihat Kode Sam [pel dan Perpustakaan](#).

Jika Anda menggunakan SDK, Anda tidak perlu menulis kode untuk menghitung tanda tangan untuk otentikasi permintaan karena klien SDK mengotentikasi permintaan Anda dengan menggunakan kunci akses yang Anda berikan. Kecuali Anda memiliki alasan bagus untuk tidak melakukannya, Anda harus selalu menggunakan AWS SDK.

- Gunakan AWS CLI untuk melakukan panggilan API Amazon S3. Untuk informasi tentang pengaturan AWS CLI dan contoh perintah Amazon S3 lihat topik berikut:

[Siapkan AWS CLI](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

[Menggunakan Amazon S3 dengan Antar AWS muka Baris Perintah](#) di Panduan AWS Command Line Interface Pengguna.

Membuat panggilan REST API langsung

 Note

Header permintaan PUT dibatasi hingga ukuran 8 KB. Dalam header permintaan PUT, metadata yang ditentukan pengguna dibatasi hingga ukuran 2 KB. Ukuran metadata

yang ditentukan sistem diukur dengan mengambil jumlah byte dalam US-ASCII pengkodean setiap kunci dan nilai.

Jika Anda ingin membuat panggilan REST API Anda sendiri alih-alih menggunakan salah satu alternatif di atas, ada beberapa hal yang perlu diingat.

- Untuk melakukan panggilan REST API langsung dari kode Anda, buat tanda tangan menggunakan kredensial yang valid dan sertakan tanda tangan dalam permintaan Anda. Untuk informasi tentang berbagai metode otentikasi dan perhitungan tanda tangan, lihat [Permintaan Otentikasi \(AWS Versi Tanda Tangan 4\)](#).
- REST API menggunakan header HTTP standar dan kode status, sehingga browser dan toolkit standar berfungsi seperti yang diharapkan. Di beberapa area, kami menambahkan fungsionalitas ke HTTP (misalnya, kami menambahkan header untuk mendukung kontrol akses). Dalam kasus ini, kami menambahkan fungsionalitas baru dengan cara yang sesuai dengan gaya penggunaan HTTP standar. Untuk informasi selengkapnya tentang membuat permintaan, lihat [Membuat permintaan](#).

Perizinan

Anda dapat memiliki kredentif yang valid untuk mengotentikasi permintaan, tetapi kecuali Anda memiliki izin S3 dari pemilik akun atau pemilik bucket, Anda tidak dapat membuat atau mengakses sumber daya Amazon S3. Izin ini biasanya diberikan melalui [kebijakan AWS Identity and Access Management \(IAM\)](#), seperti kebijakan bucket. Misalnya, Anda harus memiliki izin untuk membuat bucket S3 atau mendapatkan objek dalam bucket. Untuk daftar lengkap izin S3, lihat [Tindakan, sumber daya, dan kunci kondisi untuk Amazon S3](#).

Untuk informasi selengkapnya tentang izin untuk operasi S3 API menurut jenis sumber daya S3, lihat Izin yang [diperlukan untuk operasi Amazon S3 API](#) di Panduan Pengguna Amazon Simple Storage Service.

Jika Anda menggunakan kredensial pengguna root Anda Akun AWS, Anda memiliki semua izin. Namun, menggunakan kredensial pengguna root tidak disarankan. Sebagai gantinya, sebaiknya buat peran AWS Identity and Access Management (IAM) di akun Anda dan mengelola izin pengguna. Untuk informasi selengkapnya, lihat Manajemen [Akun](#) ses di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Mengembangkan dengan Amazon S3

Bagian ini mencakup topik terkait developer untuk menggunakan Amazon S3. Untuk informasi lebih lanjut, tinjau topik di bawah.

Topik

- [Membuat permintaan](#)
- [Mengembangkan dengan Amazon S3 menggunakan AWS CLI](#)
- [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#)
- [Mendapatkan ID permintaan Amazon S3 untuk AWS Dukungan](#)
- [Operasi API tingkat objek Amazon S3 yang didukung untuk Tabel S3](#)

Membuat permintaan

Amazon S3 adalah layanan REST. Anda dapat mengirim permintaan ke Amazon S3 menggunakan REST API atau pustaka pembungkus AWS SDK (lihat Kode Sam [pel dan Perpustakaan](#)) yang membungkus API REST Amazon S3 yang mendasarinya, menyederhanakan tugas pemrograman Anda.

Setiap interaksi dengan Amazon S3 diautentikasi atau dilakukan secara anonim. Autentikasi adalah proses verifikasi identitas pemohon yang mencoba mengakses produk Amazon Web Services (AWS). Permintaan yang diautentikasi harus menyertakan nilai tanda tangan yang mengautentikasi pengirim permintaan. Nilai tanda tangan, sebagian, dihasilkan dari kunci akses pemohon (ID kunci AWS akses dan kunci akses rahasia). Untuk informasi selengkapnya tentang mendapatkan kunci akses, lihat [Bagaimana Cara Mendapatkan Kredensial Keamanan?](#) di Referensi Umum AWS.

Jika Anda menggunakan AWS SDK, pustaka akan menghitung tanda tangan dari kunci yang Anda berikan. Namun, jika Anda melakukan panggilan API REST langsung di aplikasi Anda, Anda harus menulis kode untuk menghitung tanda tangan dan menambahkannya ke permintaan.

Topik

- [Tentang kunci akses](#)
- [Titik akhir permintaan](#)
- [Membuat permintaan ke Amazon S3 melalui IPv6](#)

- [Membuat permintaan menggunakan AWS SDK](#)
- [Membuat permintaan menggunakan API REST](#)

Tentang kunci akses

Bagian berikut membahas jenis kunci akses yang dapat Anda gunakan untuk membuat permintaan terautentikasi.

Akun AWS kunci akses

Kunci akses akun menyediakan akses penuh ke AWS sumber daya yang dimiliki oleh akun. Berikut ini adalah contoh kunci akses:

- ID kunci akses (20 karakter, string alfanumerik). Misalnya: AKIAIOSFODNN7EXAMPLE
- Kunci akses rahasia (string 40 karakter). Misalnya: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

ID kunci akses secara unik mengidentifikasi a Akun AWS. Anda dapat menggunakan kunci akses ini untuk mengirim permintaan terautentikasi ke Amazon S3.

Kunci akses pengguna IAM

Anda dapat membuat satu Akun AWS untuk perusahaan Anda; Namun, mungkin ada beberapa karyawan dalam organisasi yang membutuhkan akses ke AWS sumber daya organisasi Anda. Berbagi kunci Akun AWS akses Anda mengurangi keamanan, dan membuat individu Akun AWS untuk setiap karyawan mungkin tidak praktis. Selain itu, Anda tidak dapat membagikan sumber daya seperti bucket dan objek dengan mudah karena itu dimiliki oleh akun yang berbeda. Untuk berbagi sumber daya, Anda harus memberikan izin, yang merupakan pekerjaan tambahan.

Dalam skenario seperti itu, Anda dapat menggunakan AWS Identity and Access Management (IAM) untuk membuat pengguna di bawah Anda Akun AWS dengan kunci akses mereka sendiri dan melampirkan kebijakan pengguna IAM yang memberikan izin akses sumber daya yang sesuai kepada pengguna ini. Untuk mengelola pengguna ini dengan lebih baik, IAM memungkinkan Anda membuat kelompok pengguna dan memberikan izin tingkat kelompok yang berlaku bagi semua pengguna dalam grup tersebut.

Pengguna ini disebut sebagai pengguna IAM yang Anda buat dan kelola di dalamnya AWS. Akun induk mengontrol kemampuan pengguna untuk mengakses AWS. Setiap sumber daya yang dibuat

pengguna IAM dikendalikan dan dibayar oleh Akun AWS induk. Pengguna IAM ini dapat mengirimkan permintaan terautentikasi ke Amazon S3 menggunakan kredensial keamanan mereka sendiri. Untuk informasi selengkapnya tentang membuat dan mengelola pengguna di bawah Anda Akun AWS, buka halaman detail [AWS Identity and Access Management produk](#).

Kredensial keamanan sementara

Selain membuat pengguna IAM dengan kunci akses mereka sendiri, IAM juga memungkinkan Anda memberikan kredentif keamanan sementara (kunci akses sementara dan token keamanan) kepada pengguna IAM mana pun sehingga mereka dapat mengakses AWS layanan dan sumber daya Anda. Anda juga dapat mengelola pengguna dalam sistem Anda di luar AWS. Ini disebut sebagai pengguna gabungan. Selain itu, pengguna dapat menjadi aplikasi yang Anda buat untuk mengakses AWS sumber daya Anda.

IAM menyediakan AWS Security Token Service API bagi Anda untuk meminta kredensial keamanan sementara. Anda dapat menggunakan AWS STS API atau AWS SDK untuk meminta kredensial ini. API mengembalikan kredensial keamanan sementara (ID kunci akses dan kunci akses rahasia), dan token keamanan. Kredensial ini hanya valid untuk durasi yang Anda tentukan saat Anda memintanya. Anda menggunakan ID kunci akses dan kunci rahasia dengan cara yang sama ketika mengirim permintaan menggunakan Akun AWS Anda atau kunci akses pengguna IAM Anda. Selain itu, Anda harus menyertakan token dalam setiap permintaan yang Anda kirim ke Amazon S3.

Pengguna IAM dapat meminta kredensial keamanan sementara ini untuk penggunaan mereka sendiri atau memberikannya kepada aplikasi atau pengguna gabungan. Saat meminta kredensial keamanan sementara untuk pengguna gabungan, Anda harus memberikan nama pengguna dan kebijakan IAM yang menentukan izin yang ingin Anda kaitkan dengan kredensial keamanan sementara ini. Pengguna gabungan tidak dapat memperoleh izin lebih dari pengguna IAM induk yang meminta kredensial sementara.

Anda dapat menggunakan kredensial keamanan sementara ini dalam mengajukan permintaan kepada Amazon S3. Pustaka API menghitung nilai tanda tangan yang diperlukan menggunakan kredensial tersebut untuk mengautentikasi permintaan Anda. Jika Anda mengirim permintaan menggunakan kredensial kedaluwarsa, Amazon S3 menolak permintaan tersebut.

Untuk informasi tentang penandatanganan permintaan menggunakan kredensial keamanan sementara di permintaan API REST Anda, lihat [Menandatangani dan mengotentikasi permintaan REST \(AWS tanda tangan versi 2\)](#). Untuk informasi tentang mengirim permintaan menggunakan AWS SDK, lihat [Membuat permintaan menggunakan AWS SDK](#).

Untuk informasi selengkapnya tentang dukungan IAM untuk kredensial keamanan sementara, lihat [Kredensial Keamanan Sementara](#) dalam Panduan Pengguna IAM.

Untuk keamanan tambahan, Anda dapat mengharuskan autentikasi multi-faktor (MFA) ketika mengakses sumber daya Amazon S3 Anda dengan mengonfigurasi kebijakan bucket. Untuk selengkapnya, lihat [Contoh kebijakan bucket: Memerlukan MFA](#). Setelah Anda mengharuskan MFA untuk mengakses sumber daya Amazon S3, satu-satunya cara Anda untuk mengakses sumber daya ini adalah dengan memberikan kredensial sementara yang dibuat dengan kunci MFA. Untuk informasi selengkapnya, lihat halaman [AWS Multi-Factor](#) detail Otentikasi dan [Mengkonfigurasi Akses MFA-Protected API](#) di Panduan Pengguna IAM.

Titik akhir permintaan

Anda mengirimkan permintaan REST ke titik akhir layanan yang telah ditentukan sebelumnya. Untuk daftar semua AWS layanan dan titik akhir yang sesuai, buka [Wilayah dan Titik Akhir](#) di Referensi Umum AWS.

Membuat permintaan ke Amazon S3 melalui IPv6

Amazon Simple Storage Service (Amazon S3) mendukung kemampuan untuk mengakses bucket S3 menggunakan Protokol Internet versi 6 (IPv6), selain protokol IPv4. Titik akhir tumpukan ganda Amazon S3 mendukung permintaan ke bucket S3 melalui IPv6 dan IPv4. Tidak ada biaya tambahan untuk mengakses Amazon S3 melalui IPv6. Untuk informasi selengkapnya tentang harga, lihat [Harga Amazon S3](#).

Note

Informasi berikut adalah untuk titik akhir IPv6 Amazon S3 publik. Untuk informasi tentang titik akhir AWS Virtual Private Cloud (VPC), lihat [AWS PrivateLink untuk Amazon S3](#).

Topik

- [Memulai membuat permintaan melalui IPv6](#)
- [Menggunakan alamat IPv6 di kebijakan IAM](#)
- [Menguji kompatibilitas alamat IP](#)
- [Menggunakan titik akhir tumpukan ganda Amazon S3](#)

Memulai membuat permintaan melalui IPv6

Untuk mengajukan permintaan ke bucket S3 melalui IPv6, Anda perlu menggunakan titik akhir tumpukan ganda. Bagian selanjutnya menjelaskan cara untuk membuat permintaan melalui IPv6 dengan menggunakan titik akhir tumpukan ganda.

Berikut adalah beberapa hal yang harus Anda ketahui sebelum mencoba mengakses bucket melalui IPv6:

- Klien dan jaringan yang mengakses bucket harus diaktifkan agar dapat menggunakan IPv6.
- Permintaan cara hosting virtual dan cara jejak didukung untuk akses IPv6. Untuk informasi selengkapnya, lihat [Titik akhir tumpukan ganda Amazon S3](#).
- Jika Anda menggunakan pemfilteran alamat IP sumber dalam kebijakan pengguna atau bucket AWS Identity and Access Management (IAM), Anda perlu memperbarui kebijakan untuk menyertakan rentang alamat IPv6. Untuk informasi selengkapnya, lihat [Menggunakan alamat IPv6 di kebijakan IAM](#).
- Saat menggunakan IPv6, berkas log akses server mengeluarkan alamat IP dalam format IPv6. Anda perlu memperbarui alat, skrip, dan perangkat lunak yang ada yang Anda gunakan untuk mengurai file log Amazon S3 sehingga mereka dapat mengurai alamat Remote IP yang diformat IPv6. Untuk informasi selengkapnya, lihat [Permintaan pencatatan dengan pencatatan akses server](#).

Note

Jika Anda mengalami masalah terkait dengan kehadiran alamat IPv6 dalam berkas log, hubungi [AWS Dukungan](#).

Membuat permintaan melalui IPv6 dengan menggunakan titik akhir tumpukan ganda

Anda membuat permintaan dengan panggilan API Amazon S3 melalui IPv6 dengan menggunakan titik akhir tumpukan ganda. Operasi API Amazon S3 bekerja dengan cara yang sama baik saat Anda mengakses Amazon S3 melalui IPv6 atau melalui IPv4. Begitupun dengan kinerja.

Saat menggunakan API REST, Anda mengakses langsung titik akhir tumpukan ganda. Untuk informasi selengkapnya, lihat [Dual-stack titik akhir](#).

Saat menggunakan AWS Command Line Interface (AWS CLI) dan AWS SDK, Anda dapat menggunakan parameter atau flag untuk mengubah ke titik akhir tumpukan ganda. Anda juga dapat menentukan titik akhir tumpukan ganda secara langsung sebagai pembatalan titik akhir Amazon S3 dalam berkas config.

Anda dapat menggunakan titik akhir tumpukan ganda untuk mengakses bucket melalui IPv6 dari yang berikut ini:

- Itu AWS CLI, lihat [Menggunakan titik akhir dual-stack dari AWS CLI](#).
- AWS SDKs, lihat [Menggunakan titik akhir dual-stack dari AWS SDK](#).
- API REST, lihat [Membuat permintaan ke titik akhir tumpukan ganda menggunakan API REST](#).

Fitur tidak tersedia melalui IPv6

Fitur berikut saat ini tidak didukung ketika mengakses bucket S3 melalui IPv6: Hosting situs web statis dari bucket S3.

Menggunakan alamat IPv6 di kebijakan IAM

Sebelum mencoba mengakses bucket menggunakan IPv6, Anda harus memastikan bahwa setiap pengguna IAM atau kebijakan bucket S3 yang digunakan untuk pemfilteran alamat IP diperbarui untuk menyertakan rentang alamat IPv6. Kebijakan pemfilteran alamat IP yang tidak diperbarui untuk menangani alamat IPv6 dapat menyebabkan klien kehilangan atau mendapatkan akses ke bucket dengan tidak benar saat mereka mulai menggunakan IPv6. Untuk informasi selengkapnya tentang mengelola izin akses dengan IAM, lihat [Identitas dan Manajemen Akses untuk Amazon S3](#).

Kebijakan IAM yang memfilter alamat IP menggunakan [Operator Kondisi Alamat IP](#). Kebijakan bucket berikut mengidentifikasi rentang 54.240.143.* yang diizinkan alamat IPv4 dengan menggunakan operator kondisi alamat IP. Setiap alamat IP di luar rentang ini akan ditolak akses ke bucket (examplebucket). Karena semua alamat IPv6 di luar rentang yang diizinkan, kebijakan ini mencegah alamat IPv6 agar tidak dapat mengakses 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"}
}
]
```

Anda dapat memodifikasi elemen Condition kebijakan bucket yang memungkinkan rentang alamat IPv4 (54.240.143.0/24) dan IPv6 (2001:DB8:1234:5678::/64) seperti yang ditunjukkan dalam contoh berikut. Anda dapat menggunakan tipe blok Condition yang ditampilkan dalam contoh untuk memperbarui kebijakan pengguna IAM dan bucket Anda.

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

Sebelum menggunakan IPv6, Anda harus memperbarui semua kebijakan pengguna dan bucket IAM yang relevan yang menggunakan pemfilteran alamat IP. Hindari menggunakan pemfilteran alamat IP dalam kebijakan bucket.

Anda dapat meninjau kebijakan pengguna IAM menggunakan konsol IAM di <https://console.aws.amazon.com/iam/>. Untuk informasi lebih lanjut tentang IAM, lihat [Panduan Pengguna IAM](#). Untuk informasi tentang mengedit kebijakan bucket S3, lihat [Menambahkan kebijakan bucket](#).

Menguji kompatibilitas alamat IP

Jika Anda menggunakan use Linux/Unix atau Mac OS X, Anda dapat menguji apakah Anda dapat mengakses titik akhir tumpukan ganda melalui IPv6 dengan menggunakan `curl` perintah seperti yang ditunjukkan dalam contoh berikut:

Example

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

Anda mendapatkan informasi yang serupa dengan contoh berikut. Jika Anda terhubung melalui IPv6, alamat IP yang terhubung akan menjadi alamat 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
```

Jika Anda menggunakan Microsoft 7 atau 10, Anda dapat menguji apakah Anda dapat mengakses titik akhir dual-stack melalui IPv6 atau IPv4 dengan menggunakan ping perintah seperti yang ditunjukkan pada contoh berikut.

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

Menggunakan titik akhir tumpukan ganda Amazon S3

Titik akhir tumpukan ganda Amazon S3 mendukung permintaan ke bucket S3 melalui IPv6 dan IPv4. Bagian ini menjelaskan cara untuk menggunakan titik akhir tumpukan ganda.

Topik

- [Titik akhir tumpukan ganda Amazon S3](#)
- [Menggunakan titik akhir dual-stack dari AWS CLI](#)
- [Menggunakan titik akhir dual-stack dari AWS SDK](#)
- [Menggunakan titik akhir tumpukan ganda dari API REST](#)

Titik akhir tumpukan ganda Amazon S3

Saat Anda membuat permintaan ke titik akhir tumpukan ganda, URL bucket memutuskan ke alamat IPv6 atau IPv4. Untuk informasi lebih lanjut tentang mengakses bucket melalui IPv6, lihat [Membuat permintaan ke Amazon S3 melalui IPv6](#).

Saat menggunakan API REST, Anda secara langsung mengakses titik akhir Amazon S3 dengan menggunakan nama titik akhir (URI). Anda dapat mengakses bucket S3 melalui titik akhir tumpukan ganda dengan menggunakan nama titik akhir cara hosting virtual dan cara jejak. Amazon S3 hanya mendukung nama titik akhir tumpukan ganda regional, yang berarti Anda harus menentukan wilayah sebagai bagian dari nama.

Gunakan konvensi penamaan berikut untuk nama titik akhir cara hosting virtual dan cara jejak:

- Titik akhir tumpukan ganda cara hosting virtual:

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

- Path-style titik akhir tumpukan ganda:

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

Untuk informasi selengkapnya, tentang gaya nama titik akhir, lihat [Mengakses dan mencantumkan bucket Amazon S3](#). Untuk daftar titik akhir Amazon S3, lihat [Wilayah dan Titik Akhir](#) di Referensi Umum AWS.

Important

Anda dapat menggunakan akselerasi transfer dengan titik akhir tumpukan ganda. Untuk informasi selengkapnya, lihat [Memulai Akselerasi Transfer Amazon S3](#).

Note

Dua jenis titik akhir Virtual Private Cloud (VPC) yang mengakses Amazon S3 (titik akhir Interface VPC dan titik akhir Gateway VPC) sekarang memiliki dukungan dual-stack. Untuk informasi selengkapnya tentang titik akhir VPC untuk Amazon S3, lihat [AWS PrivateLink untuk Amazon S3](#).

Saat menggunakan AWS Command Line Interface (AWS CLI) dan AWS SDK, Anda dapat menggunakan parameter atau flag untuk mengubah ke titik akhir tumpukan ganda. Anda juga dapat menentukan titik akhir tumpukan ganda secara langsung sebagai pembatalan titik akhir Amazon S3

dalam berkas config. Bagian berikut menjelaskan cara menggunakan titik akhir dual-stack dari AWS CLI dan SDK. AWS

Menggunakan titik akhir dual-stack dari AWS CLI

Bagian ini memberikan contoh AWS CLI perintah yang digunakan untuk membuat permintaan ke titik akhir tumpukan ganda. Untuk petunjuk tentang pengaturan AWS CLI, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS CLI](#).

Anda menetapkan nilai konfigurasi `use_dualstack_endpoint` ke `true` dalam profil di AWS Config file Anda untuk mengarahkan semua permintaan Amazon S3 yang dibuat oleh `s3api` AWS CLI perintah `s3` and ke titik akhir tumpukan ganda untuk wilayah yang ditentukan. Anda menentukan wilayah dalam berkas config atau dalam perintah menggunakan opsi `--region`.

Saat menggunakan titik akhir dual-stack dengan AWS CLI, keduanya path dan gaya virtual pengalamatan didukung. Cara pengalamatan, diatur dalam berkas config, mengontrol apakah nama bucket ada dalam nama host atau bagian dari URL. Secara default, CLI akan mencoba menggunakan cara virtual jika memungkinkan, tetapi akan kembali ke cara jejak jika perlu. Untuk informasi selengkapnya, lihat [Konfigurasi Amazon S3 AWS CLI](#).

Anda juga dapat membuat perubahan konfigurasi dengan menggunakan perintah, seperti ditunjukkan dalam contoh berikut, yang menetapkan `use_dualstack_endpoint` ke `true` dan `addressing_style` ke `virtual` di profil default.

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

Jika Anda ingin menggunakan titik akhir tumpukan ganda hanya untuk AWS CLI perintah tertentu (tidak semua perintah), Anda dapat menggunakan salah satu metode berikut:

- Anda dapat menggunakan titik akhir tumpukan ganda sesuai perintah dengan mengatur parameter `--endpoint-url` ke `https://s3.dualstack.aws-region.amazonaws.com` atau `http://s3.dualstack.aws-region.amazonaws.com` untuk setiap `s3` atau perintah `s3api`.

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

- Anda dapat mengatur profil terpisah di AWS Config file Anda. Sebagai contoh, buat satu profil yang menetapkan `use_dualstack_endpoint` ke `true` dan profil yang tidak mengatur `use_dualstack_endpoint`. Ketika Anda menjalankan perintah, tentukan profil mana yang ingin

Anda gunakan, tergantung pada apakah Anda ingin menggunakan titik akhir tumpukan ganda atau tidak.

 Note

Saat menggunakan, saat ini AWS CLI Anda tidak dapat menggunakan akselerasi transfer dengan titik akhir tumpukan ganda. Namun, dukungan untuk AWS CLI itu akan segera datang. Untuk informasi selengkapnya, lihat [Mengaktifkan dan menggunakan S3 Transfer Acceleration](#).

Menggunakan titik akhir dual-stack dari AWS SDK

Bagian ini memberikan contoh cara mengakses titik akhir tumpukan ganda dengan menggunakan AWS SDK.

AWS SDK untuk Java contoh titik akhir tumpukan ganda

Contoh berikut menunjukkan cara mengaktifkan titik akhir tumpukan ganda saat membuat klien Amazon S3 menggunakan AWS SDK untuk Java.

Untuk petunjuk tentang membuat dan menguji sampel Java yang berfungsi, lihat [Memulai](#) di Panduan Peng AWS SDK untuk Java embang.

```
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();
}
}
```

Jika Anda menggunakan AWS SDK untuk Java pada Windows, Anda mungkin harus mengatur properti mesin virtual Java (JVM) berikut:

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

AWS Contoh titik akhir tumpukan ganda .NET SDK

Saat menggunakan AWS SDK untuk .NET, Anda menggunakan `AmazonS3Config` kelas untuk mengaktifkan penggunaan titik akhir tumpukan ganda seperti yang ditunjukkan pada contoh berikut.

```
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());  
        }  
    }  
}
```

Untuk informasi tentang menyiapkan dan menjalankan contoh kode, lihat Mem [ulai dengan AWS SDK untuk.NET](#) di Panduan Pengembang AWS SDK untuk.NET.

Menggunakan titik akhir tumpukan ganda dari API REST

Untuk informasi tentang permintaan sampai ke titik akhir tumpukan ganda dengan menggunakan API REST, lihat [Membuat permintaan ke titik akhir tumpukan ganda menggunakan API REST](#).

Membuat permintaan menggunakan AWS SDK

Topik

- [Membuat permintaan menggunakan Akun AWS atau kredentif pengguna IAM](#)
- [Membuat permintaan menggunakan kredensial sementara pengguna IAM](#)
- [Membuat permintaan menggunakan kredensial sementara pengguna gabungan](#)

Anda dapat mengirim permintaan yang diautentikasi ke Amazon S3 menggunakan AWS SDK atau dengan melakukan panggilan REST API langsung di aplikasi Anda. AWS SDK API menggunakan kredentif yang Anda berikan untuk menghitung tanda tangan untuk otentikasi. Jika Anda menggunakan API REST secara langsung di aplikasi Anda, Anda harus menulis kode yang diperlukan untuk menghitung tanda tangan untuk mengautentikasi permintaan Anda. Untuk daftar SDK yang tersedia, AWS buka, Kode Sam [pel dan Perpustakaan](#).

Membuat permintaan menggunakan Akun AWS atau kredentif pengguna IAM

Anda dapat menggunakan kredentif keamanan pengguna Akun AWS atau IAM untuk mengirim permintaan yang diautentikasi ke Amazon S3. Bagian ini memberikan contoh bagaimana Anda dapat mengirim permintaan yang diautentikasi menggunakan AWS SDK untuk Java, AWS SDK untuk .NET, dan AWS SDK untuk PHP. Untuk daftar SDK yang tersedia, AWS buka Kode Sampel [dan Perpustakaan](#).

Masing-masing AWS SDK ini menggunakan rantai penyedia SDK-specific kredensial untuk menemukan dan menggunakan kredensial serta melakukan tindakan atas nama pemilik kredensial. Kesamaan semua rantai penyedia kredensial ini adalah bahwa mereka semua mencari file AWS kredensial lokal Anda.

Untuk informasi selengkapnya, lihat topik di bawah:

Topik

- [Untuk membuat lokal AWS file kredensial](#)
- [Mengirim permintaan yang diautentikasi menggunakan AWS SDK](#)
- [Sumber daya terkait](#)

Untuk membuat lokal AWS file kredenSIAL

Cara termudah untuk mengonfigurasi kredenSIAL untuk AWS SDK Anda adalah dengan menggunakan file AWS kredenSIAL. Jika Anda menggunakan AWS Command Line Interface (AWS CLI), Anda mungkin sudah memiliki file AWS kredenSIAL lokal yang dikonfigurasi. Jika tidak, gunakan prosedur berikut untuk mengatur file kredensial:

1. Masuk ke Konsol Manajemen AWS dan buka konsol IAM di <https://console.aws.amazon.com/iam/>.
2. Buat pengguna baru dengan izin yang terbatas pada layanan dan tindakan yang Anda inginkan untuk dapat diakses oleh kode Anda. Untuk informasi selengkapnya tentang membuat pengguna baru, lihat [Membuat pengguna IAM \(Konsol\)](#), dan ikuti instruksi hingga langkah 8.
3. Pilih Unduh .csv untuk menyimpan salinan lokal kredensial AWS Anda.
4. Pada komputer Anda, navigasi ke direktori beranda Anda, dan buat direktori `.aws`. Pada Unix-based sistem, seperti Linux atau OS X, ini ada di lokasi berikut:

```
~/ .aws
```

Di Windows, ini ada di lokasi berikut:

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

5. Di direktori `.aws`, buat file baru bernama `credentials`.
6. Buka file `.csv` kredensial yang Anda unduh dari konsol IAM, dan salin isinya ke file `credentials` menggunakan format berikut:

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

7. Simpan file `credentials`, dan hapus file `.csv` yang Anda unduh di langkah 3.

File kredensial bersama Anda sekarang dikonfigurasi di komputer lokal Anda, dan siap digunakan dengan AWS SDK.

Mengirim permintaan yang diautentikasi menggunakan AWS SDK

Gunakan AWS SDK untuk mengirim permintaan yang diautentikasi. Untuk informasi selengkapnya tentang mengirim permintaan terautentikasi, lihat [AWS Kredensial keamanan](#) atau [Autentikasi Pusat Identitas IAM](#).

Java

Untuk informasi tentang mengotentikasi permintaan menggunakan AWS SDK untuk Java, lihat [Menggunakan file konfigurasi dan kredensial bersama untuk mengonfigurasi AWS SDK dan alat secara global serta Oten tikasi dan akses menggunakan AWS SDK dan alat](#) di Panduan Referensi AWS SDK dan Alat.

.NET

Untuk mengirim permintaan yang diautentikasi menggunakan kredensi pengguna Anda Akun AWS atau IAM:

- Buat instans dari kelas `AmazonS3Client`.
- Jalankan salah satu metode `AmazonS3Client` untuk mengirim permintaan ke Amazon S3. Klien menghasilkan tanda tangan yang diperlukan dari kredensial yang Anda berikan dan menyertakannya dalam permintaan yang dikirim ke Amazon S3.

Untuk informasi selengkapnya, lihat [Membuat permintaan menggunakan kredensi pengguna AWS akun atau IAM](#) >.

Note

- Anda dapat membuat klien `AmazonS3Client` tanpa memberikan kredensial keamanan Anda. Permintaan yang dikirim menggunakan klien ini adalah permintaan anonim, tanpa tanda tangan. Amazon S3 menampilkan pesan kesalahan jika Anda mengirim permintaan anonim untuk sumber daya yang tidak tersedia untuk umum.
- Anda dapat membuat Akun AWS dan membuat pengguna yang diperlukan. Anda juga dapat mengelola kredensial untuk pengguna tersebut. Anda perlu kredensial ini untuk melakukan tugas dalam contoh berikut. Untuk informasi selengkapnya, lihat [Mengonfigurasi kredensial AWS](#) di Panduan Developer SDK untuk .NET .

Anda kemudian juga dapat mengonfigurasi aplikasi Anda untuk secara aktif mengambil profil dan kredensial, dan kemudian secara eksplisit menggunakan kredensial

tersebut saat membuat klien AWS layanan. Untuk informasi selengkapnya, lihat [Mengakses kredensial dan profil dalam aplikasi](#) di Panduan Developer SDK untuk .NET

Contoh C# berikut menunjukkan cara melakukan tugas sebelumnya. Untuk informasi tentang menyiapkan dan menjalankan contoh kode, lihat [Memulai dengan AWS SDK untuk .NET](#) di Panduan Pengembang AWS SDK untuk .NET.

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

Bagian ini menjelaskan cara menggunakan kelas dari versi 3 AWS SDK untuk PHP untuk mengirim permintaan yang diautentikasi menggunakan kredentif pengguna Anda Akun AWS atau

IAM. Untuk informasi selengkapnya tentang AWS SDK untuk Ruby API, buka [AWS SDK untuk Ruby - Versi 2](#).

Contoh PHP berikut menunjukkan bagaimana klien mengajukan permintaan menggunakan kredensial keamanan Anda untuk mencantumkan semua bucket untuk akun Anda.

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

Anda dapat membuat klien `S3Client` tanpa memberikan kredensial keamanan Anda. Permintaan yang dikirim menggunakan klien ini adalah permintaan anonim, tanpa tanda tangan. Amazon S3 menampilkan pesan kesalahan jika Anda mengirim permintaan anonim untuk sumber daya yang tidak tersedia untuk umum. Untuk informasi selengkapnya, lihat [Membuat Klien Anonim](#) dalam [Dokumentasi AWS SDK untuk PHP](#).

Ruby

Sebelum Anda dapat menggunakan versi 3 AWS SDK untuk Ruby untuk melakukan panggilan ke Amazon S3, Anda harus mengatur kredentif AWS akses yang digunakan SDK untuk memverifikasi akses Anda ke bucket dan objek. Jika Anda telah menyiapkan kredensial bersama di profil AWS kredensial di sistem lokal Anda, versi 3 SDK untuk Ruby dapat menggunakan kredensial tersebut tanpa Anda harus mendeklarasikannya dalam kode Anda. Untuk informasi selengkapnya tentang menyiapkan kredensial bersama, lihat [Membuat permintaan menggunakan kredensial](#) pengguna AWS akun atau IAM.

Cuplikan kode Ruby berikut menggunakan kredensial dalam file AWS kredensial bersama di komputer lokal untuk mengotentikasi permintaan untuk mendapatkan semua nama kunci objek dalam bucket tertentu. Contoh ini melakukan hal-hal berikut:

1. Buat instans dari kelas `Aws::S3::Client`.
2. Membuat permintaan ke Amazon S3 dengan menghitung objek dalam bucket menggunakan metode `list_objects_v2` dari `Aws::S3::Client`. Klien menghasilkan nilai tanda tangan yang diperlukan dari kredensial dalam file AWS kredensial di komputer Anda, dan memasukkannya ke dalam permintaan yang dikirimkannya ke Amazon S3.
3. Mencetak susunan nama kunci objek ke terminal.

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__

```

Jika Anda tidak memiliki file AWS kredensial lokal, Anda masih dapat membuat `Aws::S3::Client` sumber daya dan menjalankan kode terhadap bucket dan objek Amazon S3. Permintaan yang dikirim menggunakan SDK versi 3 untuk Ruby bersifat anonim, tanpa tanda

tangan default. Amazon S3 menampilkan pesan kesalahan jika Anda mengirim permintaan anonim untuk sumber daya yang tidak tersedia untuk umum.

Anda dapat menggunakan dan memperluas potongan kode sebelumnya untuk SDK untuk aplikasi Ruby, seperti dalam contoh yang lebih kuat berikut.

```
# 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

Contoh berikut menggunakan AWS kredensial yang dimuat secara otomatis oleh SDK untuk Go dari file kredensial bersama.

```
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)
    }
}
}
```

Sumber daya terkait

- [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#)
- [AWS SDK untuk PHP untuk Aws\S3\S3Client Class Amazon S3](#)
- [Dokumentasi AWS SDK untuk PHP](#)

Membuat permintaan menggunakan kredensial sementara pengguna IAM

Pengguna Akun AWS atau pengguna IAM dapat meminta kredentif keamanan sementara dan menggunakannya untuk mengirim permintaan yang diautentikasi ke Amazon S3. Bagian ini memberikan contoh cara menggunakan AWS SDK untuk Java, .NET, dan PHP untuk memperoleh kredensial keamanan sementara dan menggunakannya untuk mengautentikasi permintaan Anda ke Amazon S3.

Java

Pengguna IAM atau pengguna Akun AWS dapat meminta kredensial keamanan sementara (lihat [Membuat permintaan](#)) menggunakan AWS SDK untuk Java dan menggunakannya untuk mengakses Amazon S3. Kredensial ini berakhir setelah durasi sesi yang ditentukan.

Secara default, durasi sesi adalah satu jam. Jika Anda menggunakan kredensial pengguna IAM, Anda dapat menentukan durasi saat meminta kredensial keamanan sementara dari 15 menit hingga durasi sesi maksimum untuk peran tersebut. Untuk informasi selengkapnya tentang kredensial keamanan sementara, lihat [Kredensial Keamanan Sementara](#) dalam Panduan Pengguna IAM. Untuk informasi selengkapnya tentang membuat permintaan, lihat [Membuat permintaan](#).

Untuk mendapatkan kredensial keamanan sementara dan mengakses Amazon S3

1. Buat instans dari kelas `AWSSecurityTokenService`.
2. Ambil kredensial keamanan sementara untuk peran yang diinginkan dengan memanggil metode `assumeRole()` klien Security Token Service (STS).
3. Kemas kredensial keamanan sementara menjadi objek `BasicSessionCredentials`. Anda menggunakan objek ini untuk menyediakan kredensial keamanan sementara kepada klien Amazon S3.
4. Buat instans kelas `AmazonS3Client` dengan menggunakan kredensial keamanan sementara. Anda mengirim permintaan ke Amazon S3 menggunakan klien ini. Jika Anda mengirim permintaan menggunakan kredensial kedaluwarsa, Amazon S3 akan menampilkan pesan kesalahan.

Note

Jika Anda memperoleh kredensial keamanan sementara menggunakan kredensial keamanan Akun AWS Anda, kredensial sementara hanya berlaku selama satu jam. Anda

dapat menentukan durasi sesi hanya jika Anda menggunakan kredensial pengguna IAM untuk meminta sesi.

Contoh berikut ini mencantumkan serangkaian kunci objek dalam bucket yang ditentukan. Contoh tersebut memperoleh kredensial keamanan sementara untuk suatu sesi dan menggunakannya untuk mengirim permintaan terautentikasi ke Amazon S3.

Jika Anda ingin menguji sampel menggunakan kredensial pengguna IAM, Anda harus membuat pengguna IAM di bawah Akun AWS Anda. Untuk informasi selengkapnya tentang cara membuat pengguna IAM, lihat [Membuat Grup Pengguna dan Administrator IAM Pertama Anda](#) dalam Panduan Pengguna IAM.

Untuk petunjuk tentang membuat dan menguji sampel yang berfungsi, lihat [Memulai](#) di Panduan Peng AWS SDK untuk Java embang.

```
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

Pengguna IAM atau pengguna Akun AWS dapat meminta kredensial keamanan sementara menggunakan AWS SDK untuk .NET dan menggunakannya untuk mengakses Amazon S3. Kredensial ini berakhir setelah durasi sesi.

Secara default, durasi sesi adalah satu jam. Jika Anda menggunakan kredensial pengguna IAM, Anda dapat menentukan durasi saat meminta kredensial keamanan sementara dari 15 menit hingga durasi sesi maksimum untuk peran tersebut. Untuk informasi selengkapnya tentang kredensial keamanan sementara, lihat [Kredensial Keamanan Sementara](#) dalam Panduan Pengguna IAM. Untuk informasi selengkapnya tentang membuat permintaan, lihat [Membuat permintaan](#).

Untuk mendapatkan kredensial keamanan sementara dan mengakses Amazon S3

1. Buat instance AWS Security Token Service klien, `AmazonSecurityTokenServiceClient`.
2. Mulai sesi dengan memanggil metode `GetSessionToken` klien STS yang Anda buat pada langkah sebelumnya. Anda memberikan informasi sesi untuk metode ini menggunakan objek `GetSessionTokenRequest`.

Metode tersebut mengembalikan kredensial keamanan sementara Anda.

3. Kemas kredensial keamanan sementara dalam instans objek `SessionAWSCredentials`. Anda menggunakan objek ini untuk menyediakan kredensial keamanan sementara kepada klien Amazon S3.
4. Buat instans kelas `AmazonS3Client` dengan memasukkan kredensial keamanan sementara. Anda mengirim permintaan ke Amazon S3 menggunakan klien ini. Jika Anda mengirim permintaan menggunakan kredensial kedaluwarsa, Amazon S3 akan menampilkan pesan kesalahan.

 Note

Jika Anda memperoleh kredensial keamanan sementara menggunakan kredensial keamanan Akun AWS Anda, kredensial sementara hanya berlaku selama satu jam. Anda dapat menentukan durasi sesi hanya jika Anda menggunakan kredensial pengguna IAM untuk meminta sesi.

Contoh C# berikut mencantumkan kunci objek dalam bucket yang ditentukan. Sebagai ilustrasi, contoh tersebut memperoleh kredensial keamanan sementara untuk sesi default satu jam dan menggunakannya untuk mengirim permintaan terautentikasi ke Amazon S3.

Jika Anda ingin menguji sampel menggunakan kredensial pengguna IAM, Anda harus membuat pengguna IAM di bawah Akun AWS Anda. Untuk informasi selengkapnya tentang cara membuat pengguna IAM, lihat [Membuat Grup Pengguna dan Administrator IAM Pertama Anda](#) dalam Panduan Pengguna IAM. Untuk informasi selengkapnya tentang membuat permintaan, lihat [Membuat permintaan](#).

Untuk informasi tentang menyiapkan dan menjalankan contoh kode, lihat [Memulai dengan AWS SDK untuk .NET](#) di Panduan Pengembang AWS SDK untuk .NET.

```
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

Untuk informasi selengkapnya tentang AWS SDK untuk Ruby API, buka [AWS SDK untuk Ruby - Versi 2](#).

Pengguna IAM atau pengguna Akun AWS dapat meminta kredensial keamanan sementara menggunakan versi 3 dari AWS SDK untuk PHP. Itu kemudian dapat menggunakan kredensial sementara untuk mengakses Amazon S3. Kredensial berakhir saat durasi sesi berakhir.

Secara default, durasi sesi adalah satu jam. Jika Anda menggunakan kredensial pengguna IAM, Anda dapat menentukan durasi saat meminta kredensial keamanan sementara dari 15 menit hingga durasi sesi maksimum untuk peran tersebut. Untuk informasi selengkapnya tentang kredensial keamanan sementara, lihat [Kredensial Keamanan Sementara](#) dalam Panduan Pengguna IAM. Untuk informasi selengkapnya tentang membuat permintaan, lihat [Membuat permintaan](#).

Note

Jika Anda memperoleh kredensial keamanan sementara menggunakan kredensial keamanan Akun AWS Anda, kredensial keamanan sementara hanya berlaku selama

satu jam. Anda dapat menentukan durasi sesi hanya jika Anda menggunakan kredensial pengguna IAM untuk meminta sesi.

Example

Contoh PHP berikut ini mencantumkan kunci objek dalam bucket tertentu menggunakan kredensial keamanan sementara. Contoh tersebut memperoleh kredensial keamanan sementara untuk sesi satu jam default, dan menggunakannya untuk mengirim permintaan terautentikasi ke Amazon S3. Untuk informasi selengkapnya tentang AWS SDK untuk Ruby API, buka [AWS SDK untuk Ruby - Versi 2](#).

Jika Anda ingin menguji contoh menggunakan kredensial pengguna IAM, Anda harus membuat pengguna IAM di bawah Akun AWS Anda. Untuk informasi tentang cara membuat pengguna IAM, lihat [Membuat Grup Pengguna dan Administrator IAM Pertama Anda](#) dalam Panduan Pengguna IAM. Untuk contoh dari pengaturan durasi sesi saat menggunakan kredensial pengguna IAM untuk meminta sesi, lihat [Membuat permintaan menggunakan kredensial sementara pengguna 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

Pengguna IAM atau pengguna Akun AWS dapat meminta kredensial keamanan sementara menggunakan AWS SDK untuk Ruby dan menggunakannya untuk mengakses Amazon S3. Kredensial ini berakhir setelah durasi sesi.

Secara default, durasi sesi adalah satu jam. Jika Anda menggunakan kredensial pengguna IAM, Anda dapat menentukan durasi saat meminta kredensial keamanan sementara dari 15 menit hingga durasi sesi maksimum untuk peran tersebut. Untuk informasi selengkapnya tentang kredensial keamanan sementara, lihat [Kredensial Keamanan Sementara](#) dalam Panduan Pengguna IAM. Untuk informasi selengkapnya tentang membuat permintaan, lihat [Membuat permintaan](#).

Note

Jika Anda memperoleh kredensial keamanan sementara menggunakan kredensial keamanan Akun AWS Anda, kredensial keamanan sementara hanya berlaku selama satu jam. Anda dapat menentukan durasi sesi hanya jika Anda menggunakan kredensial pengguna IAM untuk meminta sesi.

Contoh Ruby berikut membuat pengguna sementara untuk mencantumkan item dalam bucket tertentu selama satu jam. Untuk menggunakan contoh ini, Anda harus memiliki AWS kredensial yang memiliki izin yang diperlukan untuk membuat klien baru AWS Security Token Service (AWS STS), dan mencantumkan 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
```

Membuat permintaan menggunakan kredensial sementara pengguna gabungan

Anda dapat meminta kredentif keamanan sementara dan memberikannya kepada pengguna atau aplikasi federasi yang perlu mengakses AWS sumber daya Anda. Bagian ini memberikan contoh bagaimana Anda dapat menggunakan AWS SDK untuk mendapatkan kredensial keamanan sementara untuk pengguna atau aplikasi federasi Anda dan mengirim permintaan yang diautentikasi ke Amazon S3 menggunakan kredensial tersebut. Untuk daftar SDK yang tersedia, lihat [Contoh Kode dan AWS Pustaka](#).

Note

Baik pengguna Akun AWS dan pengguna IAM dapat meminta kredentif keamanan sementara untuk pengguna federasi. Namun, untuk keamanan tambahan, hanya pengguna IAM dengan izin yang diperlukan yang harus meminta kredensial sementara ini untuk memastikan bahwa pengguna gabungan mendapatkan paling banyak izin dari pengguna IAM yang meminta. Dalam beberapa aplikasi, Anda mungkin menganggapnya sesuai untuk membuat pengguna IAM dengan izin tertentu semata-mata untuk tujuan memberikan kredensial keamanan sementara kepada pengguna gabungan dan aplikasi Anda.

Java

Anda dapat memberikan kredentif keamanan sementara untuk pengguna dan aplikasi federasi sehingga mereka dapat mengirim permintaan yang diautentikasi untuk mengakses sumber daya Anda AWS . Saat meminta kredensial sementara ini, Anda harus memberikan nama pengguna dan kebijakan IAM yang menjelaskan izin sumber daya yang ingin Anda berikan. Secara default, durasi sesi adalah satu jam. Anda dapat secara eksplisit mengatur nilai durasi yang berbeda ketika meminta kredensial keamanan sementara untuk pengguna gabungan dan aplikasi.

Note

Untuk keamanan tambahan saat meminta kredensial keamanan sementara untuk pengguna gabungan dan aplikasi, kami menyarankan agar Anda menggunakan pengguna IAM khusus hanya dengan izin akses yang diperlukan. Pengguna sementara yang Anda buat tidak akan pernah mendapatkan izin lebih dari pengguna IAM yang meminta kredensial keamanan sementara. Untuk informasi selengkapnya, lihat [FAQ AWS Identity and Access Management](#).

Untuk memberikan kredensial keamanan dan mengirimkan permintaan terautentikasi untuk mengakses sumber daya, lakukan hal berikut:

- Buat instans dari kelas `AWSSecurityTokenServiceClient`.
- Mulai sesi dengan memanggil metode `getFederationToken()` klien Security Token Service (STS). Berikan informasi sesi, termasuk nama pengguna dan kebijakan IAM, yang ingin Anda lampirkan ke kredensial sementara. Anda dapat memberikan durasi sesi opsional. Metode tersebut mengembalikan kredensial keamanan sementara Anda.
- Kemas kredensial keamanan sementara dalam instans objek `BasicSessionCredentials`. Anda menggunakan objek ini untuk menyediakan kredensial keamanan sementara kepada klien Amazon S3.
- Buat instans kelas `AmazonS3Client` dengan menggunakan kredensial keamanan sementara. Anda mengirim permintaan ke Amazon S3 menggunakan klien ini. Jika Anda mengirim permintaan menggunakan kredensial kedaluwarsa, Amazon S3 akan menampilkan pesan kesalahan.

Example

Contoh ini mencantumkan kunci dalam bucket S3 yang ditentukan. Dalam contoh, Anda memperoleh kredensial keamanan sementara untuk sesi dua jam bagi pengguna gabungan Anda dan menggunakan kredensial untuk mengirim permintaan yang diautentikasi ke Amazon S3. Untuk menjalankan contoh, Anda perlu membuat pengguna IAM dengan kebijakan terlampir yang memungkinkan pengguna meminta kredensial keamanan sementara dan mencantumkan AWS sumber daya Anda. Kebijakan berikut mencapai hal ini:

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

Untuk informasi selengkapnya tentang cara membuat pengguna IAM, lihat [Membuat Grup Pengguna dan Administrator IAM Pertama Anda](#) dalam Panduan Pengguna IAM.

Setelah membuat pengguna IAM dan melampirkan kebijakannya, Anda dapat menjalankan contoh berikut. Untuk petunjuk tentang membuat dan menguji sampel yang berfungsi, lihat [Memulai](#) di Panduan Peng AWS SDK untuk Java embang.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSSessionCredentialsProvider;
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

Anda dapat memberikan kredentif keamanan sementara untuk pengguna dan aplikasi federasi sehingga mereka dapat mengirim permintaan yang diautentikasi untuk mengakses sumber daya Anda AWS . Saat meminta kredensial sementara ini, Anda harus memberikan nama pengguna dan kebijakan IAM yang menjelaskan izin sumber daya yang ingin Anda berikan. Secara default, durasi sesi adalah satu jam. Anda dapat secara eksplisit mengatur nilai durasi yang berbeda ketika meminta kredensial keamanan sementara untuk pengguna gabungan dan aplikasi. Untuk informasi tentang mengirim permintaan terautentikasi, lihat [Membuat permintaan](#).

Note

Saat meminta kredentif keamanan sementara untuk pengguna dan aplikasi federasi, untuk keamanan tambahan, gunakan pengguna IAM khusus hanya dengan izin akses yang diperlukan. Pengguna sementara yang Anda buat tidak akan pernah mendapatkan izin lebih dari pengguna IAM yang meminta kredensial keamanan sementara. Untuk informasi selengkapnya, lihat [FAQ AWS Identity and Access Management](#).

Anda melakukan hal berikut:

- Buat instance AWS Security Token Service klien, `AmazonSecurityTokenServiceClient` kelas.
- Mulai sesi dengan memanggil metode `GetFederationToken` klien STS. Anda perlu memberikan informasi sesi, termasuk nama pengguna dan kebijakan IAM yang ingin Anda lampirkan ke kredensial sementara. Atau, Anda dapat memberikan durasi sesi. Metode tersebut mengembalikan kredensial keamanan sementara Anda.
- Kemas kredensial keamanan sementara dalam instans objek `SessionAWSCredentials`. Anda menggunakan objek ini untuk menyediakan kredensial keamanan sementara kepada klien Amazon S3.
- Buat instans kelas `AmazonS3Client` dengan menyerahkan kredensial keamanan sementara. Anda menggunakan klien ini untuk mengirim permintaan ke Amazon S3. Jika Anda mengirim permintaan menggunakan kredensial kedaluwarsa, Amazon S3 akan menampilkan pesan kesalahan.

Example

Contoh C# berikut mencantumkan kunci dalam bucket yang ditentukan. Dalam contoh, Anda memperoleh kredensial keamanan sementara untuk sesi dua jam bagi pengguna gabungan (Pengguna 1) Anda dan menggunakan kredensial untuk mengirim permintaan yang diautentikasi ke Amazon S3.

- Untuk latihan ini, Anda membuat pengguna IAM dengan izin minimal. Dengan menggunakan kredensial pengguna IAM ini, Anda meminta kredensial sementara untuk orang lain. Contoh ini hanya mencantumkan objek dalam bucket tertentu. Buat pengguna IAM dengan kebijakan berikut terlampir:

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

Kebijakan ini memungkinkan pengguna IAM meminta kredentif keamanan sementara dan izin akses hanya untuk mencantumkan AWS sumber daya Anda. Untuk informasi selengkapnya tentang cara membuat pengguna IAM, lihat [Membuat Grup Pengguna dan Administrator IAM Anda](#) dalam Panduan Pengguna IAM.

- Gunakan kredensial keamanan pengguna IAM untuk menguji contoh berikut. Contoh tersebut mengirimkan permintaan terautentikasi ke Amazon S3 menggunakan kredensial keamanan sementara. Contoh tersebut menentukan kebijakan berikut ketika meminta kredensial keamanan sementara untuk pengguna gabungan (Pengguna1), yang membatasi akses untuk mencantumkan objek dalam bucket tertentu (YourBucketName). Anda harus memperbarui kebijakan dan memberikan nama bucket Anda sendiri.

```
{
  "Statement": [
    {
      "Sid": "1",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
```

```
        "Resource": "arn:aws:s3:::YourBucketName"
    }
]
}
```

- Example

Perbarui sampel berikut dan berikan nama bucket yang Anda tentukan dalam kebijakan akses pengguna gabungan sebelumnya. Untuk informasi tentang menyiapkan dan menjalankan contoh kode, lihat Mem [ulai dengan AWS SDK untuk.NET](#) di Panduan Pengembang AWS SDK untuk.NET.

```
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");
            }
            catch { }
        }
    }
}
```

```

        // 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

Topik ini menjelaskan cara menggunakan kelas dari versi 3 AWS SDK untuk PHP untuk meminta kredentif keamanan sementara untuk pengguna dan aplikasi federasi dan menggunakannya untuk mengakses sumber daya yang disimpan di Amazon S3. Untuk informasi selengkapnya tentang AWS SDK untuk Ruby API, buka [AWS SDK untuk Ruby - Versi 2](#).

Anda dapat memberikan kredentif keamanan sementara kepada pengguna dan aplikasi federasi sehingga mereka dapat mengirim permintaan yang diautentikasi untuk mengakses sumber daya Anda AWS . Saat meminta kredensial sementara ini, Anda harus memberikan nama pengguna dan kebijakan IAM yang menjelaskan izin sumber daya yang ingin Anda berikan.

Kredensial ini berakhir saat durasi sesi berakhir. Secara default, durasi sesi adalah satu jam. Anda dapat secara eksplisit mengatur nilai yang berbeda selama durasi saat meminta kredensial keamanan sementara untuk pengguna gabungan dan aplikasi. Untuk informasi selengkapnya tentang kredensial keamanan sementara, lihat [Kredensial Keamanan Sementara](#) dalam Panduan Pengguna IAM. Untuk informasi tentang penyediaan kredensial keamanan sementara bagi pengguna gabungan dan aplikasi Anda, lihat [Membuat permintaan](#).

Untuk keamanan tambahan saat meminta kredensial keamanan sementara untuk pengguna gabungan dan aplikasi, kami menyarankan untuk menggunakan pengguna IAM khusus hanya dengan izin akses yang diperlukan. Pengguna sementara yang Anda buat tidak akan pernah mendapatkan izin lebih dari pengguna IAM yang meminta kredensial keamanan sementara. Untuk informasi tentang federasi identitas, lihat [FAQ AWS Identity and Access Management](#).

Untuk informasi selengkapnya tentang AWS SDK untuk Ruby API, buka [AWS SDK untuk Ruby - Versi 2](#).

Example

Contoh PHP berikut ini mencantumkan kunci dalam bucket yang ditentukan. Dalam contoh ini, Anda memperoleh kredensial keamanan sementara untuk satu sesi jam bagi pengguna gabungan Anda (Pengguna1). Kemudian Anda menggunakan kredensial keamanan sementara untuk mengirim permintaan terautentikasi ke Amazon S3.

Untuk keamanan tambahan saat meminta kredensial sementara untuk orang lain, Anda menggunakan kredensial keamanan pengguna IAM yang memiliki izin untuk meminta kredensial keamanan sementara. Untuk memastikan bahwa pengguna IAM hanya memberikan izin spesifik aplikasi minimum kepada pengguna gabungan, Anda juga dapat membatasi izin akses pengguna IAM ini. Contoh ini hanya mencantumkan objek dalam bucket tertentu. Buat pengguna IAM dengan kebijakan berikut terlampir:

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

Kebijakan ini memungkinkan pengguna IAM meminta kredentif keamanan sementara dan izin akses hanya untuk mencantumkan AWS sumber daya Anda. Untuk informasi selengkapnya tentang cara membuat pengguna IAM, lihat [Membuat Grup Pengguna dan Administrator IAM Pertama Anda](#) dalam Panduan Pengguna IAM.

Sekarang Anda dapat menggunakan kredensial keamanan pengguna IAM untuk menguji contoh berikut. Contoh tersebut mengirimkan permintaan terautentikasi ke Amazon S3 menggunakan kredensial keamanan sementara. Ketika meminta kredensial keamanan sementara untuk pengguna gabungan (Pengguna1), contoh tersebut menentukan kebijakan berikut, yang membatasi akses ke objek daftar dalam bucket tertentu. Perbarui kebijakan dengan nama bucket Anda.

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

Dalam contoh berikut, ketika menentukan sumber daya kebijakan, ganti YourBucketName dengan nama bucket Anda.:

```
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

Anda dapat memberikan kredentif keamanan sementara untuk pengguna dan aplikasi federasi sehingga mereka dapat mengirim permintaan yang diautentikasi untuk mengakses sumber daya Anda AWS . Saat meminta kredensial sementara dari layanan IAM, Anda harus memberikan nama pengguna dan kebijakan IAM yang menjelaskan izin sumber daya yang ingin Anda berikan. Secara default, durasi sesi adalah satu jam. Namun, jika Anda meminta kredensial sementara

menggunakan kredensial pengguna IAM, Anda dapat secara eksplisit mengatur nilai durasi yang berbeda ketika meminta kredensial keamanan sementara untuk pengguna gabungan dan aplikasi. Untuk informasi tentang kredensial keamanan sementara untuk pengguna dan aplikasi gabungan Anda, lihat [Membuat permintaan](#).

Note

Untuk keamanan tambahan ketika Anda meminta kredensial keamanan sementara untuk pengguna gabungan dan aplikasi, Anda mungkin ingin menggunakan pengguna IAM khusus hanya dengan izin akses yang diperlukan. Pengguna sementara yang Anda buat tidak akan pernah mendapatkan izin lebih dari pengguna IAM yang meminta kredensial keamanan sementara. Untuk informasi selengkapnya, lihat [FAQ AWS Identity and Access Management](#).

Example

Contoh kode Ruby berikut memungkinkan pengguna gabungan dengan set izin terbatas untuk mencantumkan kunci dalam bucket tertentu.

```
# 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}"
end

```

```

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__

```

Membuat permintaan menggunakan API REST

Bagian ini berisi informasi tentang cara membuat permintaan ke titik akhir Amazon S3 dengan menggunakan API REST. Untuk daftar titik akhir Amazon S3, lihat [Wilayah dan Titik Akhir](#) di Referensi Umum AWS.

Membangun nama host S3 untuk permintaan API REST

Titik akhir Amazon S3 mengikuti struktur yang ditunjukkan di bawah ini:

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

Titik akhir titik akses Amazon S3 dan titik akhir tumpukan ganda juga mengikuti struktur standar:

- Titik akses Amazon S3 -s3-accesspoint.*Region*.amazonaws.com
- Dual-stack - s3.dualstack.*Region*.amazonaws.com

Untuk daftar lengkap titik akhir dan Wilayah Amazon S3, lihat [Titik akhir dan kuota Amazon S3](#) di Referensi Umum Amazon Web Services.

Permintaan gaya jalur dan gaya hosting virtual

Saat membuat permintaan dengan menggunakan API REST, Anda dapat menggunakan gaya jalur atau URI gaya hosting virtual untuk titik akhir Amazon S3. Untuk informasi selengkapnya, lihat [Path-style permintaan](#).

Example Permintaan gaya hosting virtual

Berikut ini adalah contoh permintaan gaya hosting virtual untuk menghapus file puppy.jpg dari bucket bernama examplebucket di Wilayah AS Barat (Oregon). Untuk informasi selengkapnya tentang permintaan gaya host virtual, lihat [Path-style permintaan](#).

```
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 permintaan

Berikut ini adalah contoh versi gaya jalur dari permintaan yang sama.

```
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
```

Saat ini, Amazon S3 mendukung akses URL dengan gaya hosting virtual dan gaya jalur di seluruh Wilayah AWS. Namun, URL cara jalur akan dihentikan di masa depan. Untuk informasi selengkapnya, lihat catatan Penting berikut ini.

Untuk informasi selengkapnya tentang permintaan gaya jalur, lihat [Path-style permintaan](#).

Important

Pembaruan (23 September 2020)–Untuk memastikan bahwa pelanggan memiliki waktu yang dibutuhkan untuk melakukan transisi ke URL dengan cara hosting virtual, kami telah

memutuskan untuk menunda penghentian penggunaan URL dengan cara jalur. Untuk informasi lebih lanjut, lihat [Rencana Penghentian Jalur Amazon S3-Cerita Seluruhnya](#) di Blog Berita AWS .

Membuat permintaan ke titik akhir tumpukan ganda menggunakan API REST

Saat menggunakan API REST, Anda dapat mengakses langsung titik akhir tumpukan ganda dengan menggunakan nama titik akhir (URI) gaya hosting virtual dan gaya jalur. Semua nama titik akhir tumpukan ganda Amazon S3 mencakup wilayah dalam nama tersebut. Tidak seperti titik akhir IPv4-only standar, titik akhir gaya host virtual dan jalur gaya menggunakan nama titik akhir khusus wilayah.

Example Permintaan titik akhir tumpukan ganda gaya hosting virtual

Anda dapat menggunakan titik akhir gaya hosting virtual dalam permintaan REST seperti yang ditunjukkan dalam contoh berikut yang mengambil objek puppy . jpg dari bucket yang bernama examplebucket di Wilayah AS Barat (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 permintaan titik akhir tumpukan ganda

Atau Anda dapat menggunakan titik akhir gaya jalur dalam permintaan Anda seperti ditunjukkan dalam contoh berikut.

```
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
```

Untuk informasi selengkapnya tentang titik akhir tumpukan ganda, lihat [Menggunakan titik akhir tumpukan ganda Amazon S3](#).

Untuk informasi selengkapnya tentang membuat permintaan menggunakan REST API, lihat topik berikut.

Topik

- [Meminta pengalihan dan API REST](#)
- [Meminta perutean](#)

Meminta pengalihan dan API REST

Topik

- [Pengalihan dan agen pengguna HTTP](#)
- [Mengalihkan dan 100-Lanjutkan](#)
- [Contoh pengalihan](#)

Bagian ini menjelaskan cara menangani pengalihan HTTP menggunakan API REST Amazon S3. Untuk informasi umum tentang pengalihan Amazon S3, lihat [Membuat permintaan](#) dalam Referensi API Amazon Simple Storage Service.

Pengalihan dan agen pengguna HTTP

Program yang menggunakan API REST Amazon S3 harus menangani pengalihan baik pada lapisan aplikasi atau lapisan HTTP. Banyak pustaka klien dan agen pengguna HTTP dapat dikonfigurasi agar menangani pengalihan secara otomatis dengan tepat; tetapi, banyak klien lain yang memiliki implementasi pengalihan yang tidak tepat atau tidak lengkap.

Sebelum Anda mengandalkan pustaka untuk memenuhi persyaratan pengalihan, lakukan uji kasus berikut:

- Verifikasi bahwa semua header permintaan HTTP disertakan dengan benar dalam permintaan pengalihan (permintaan kedua setelah menerima pengalihan) termasuk standar HTTP seperti Otorisasi dan Tanggal.
- Verifikasi pengalihan non-DAPATKAN, seperti LETAKKAN dan HAPUS, bekerja dengan benar.
- Verifikasi permintaan LETAKKAN yang besar mengikuti pengalihan dengan benar.
- Verifikasi permintaan LETAKKAN mengikuti arah yang benar jika respons 100-lanjutkan memerlukan waktu lama untuk sampai.

Agan pengguna HTTP yang secara ketat mematuhi RFC 2616 mungkin memerlukan konfirmasi eksplisit sebelum mengikuti pengalihan jika metode permintaan HTTP bukan DAPATKAN atau

HEAD. Mengikuti pengalihan yang dibuat oleh Amazon S3 secara otomatis umumnya aman, karena sistem hanya akan mengeluarkan pengalihan ke host dalam domain amazonaws.com dan efek permintaan yang dialihkan akan sama dengan permintaan asli.

Mengalihkan dan 100-Lanjutkan

Untuk menyederhanakan penanganan pengalihan, meningkatkan efisiensi, dan menghindari biaya yang terkait dengan pengiriman isi permintaan pengalihan dua kali, konfigurasi aplikasi Anda untuk menggunakan 100-lanjutkan untuk operasi LETAKKAN. Ketika aplikasi Anda menggunakan 100-lanjutkan, aplikasi tidak mengirim isi permintaan sampai menerima pengakuan. Jika pesan ditolak berdasarkan header, isi pesan tidak akan terkirim. Untuk informasi selengkapnya tentang 100-lanjutkan, kunjungi [RFC 2616 Bagian 8.2.3](#)

Note

Menurut RFC 2616, saat menggunakan Expect: Continue dengan server HTTP yang tidak dikenal, Anda tidak boleh menunggu selama periode yang tidak terbatas sebelum mengirimkan isi permintaan. Hal ini karena beberapa server HTTP tidak mengetahui adanya 100-lanjutkan. Namun, Amazon S3 mengetahui jika permintaan Anda memuat Expect: Continue dan akan merespons dengan status 100-lanjutkan sementara atau kode status akhir. Selain itu, tidak akan terjadi kesalahan pengalihan setelah menerima 100 izin lanjutkan sementara. Ini akan membantu Anda menghindari menerima respons pengalihan saat Anda masih menulis isi permintaan.

Contoh pengalihan

Bagian ini memberikan contoh interaksi server-klien menggunakan pengalihan HTTP dan 100-lanjutkan.

Berikut ini adalah contoh LETAKKAN ke 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 memberikan informasi berikut:

```
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>
```

Klien mengikuti respons pengalihan dan mengeluarkan permintaan baru ke titik akhir sementara `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 mengembalikan 100-lanjutkan yang menunjukkan bahwa klien harus melanjutkan dengan mengirim isi permintaan.

```
HTTP/1.1 100 Continue
```

Klien mengirimkan isi permintaan.

```
ha ha\n
```

Amazon S3 mengembalikan respons akhir.


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

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

Meminta perutean

Program yang membuat permintaan terhadap bucket yang dibuat menggunakan [CreateBucket](#) API yang menyertakan pengalihan [CreateBucketConfiguration](#) harus mendukung. Selain itu, beberapa klien yang tidak menghormati DNS TTL dapat menghadapi masalah.

Bagian ini menjelaskan perutean dan masalah DNS untuk dipertimbangkan saat merancang layanan atau aplikasi Anda untuk digunakan dengan Amazon S3.

Meminta pengalihan dan API REST

Amazon S3 menggunakan Sistem Nama Domain (DNS) untuk merutekan permintaan ke fasilitas yang dapat memprosesnya. Sistem ini bekerja secara efektif, tetapi kesalahan perutean sementara dapat terjadi. Jika permintaan tiba di lokasi Amazon S3 yang salah, Amazon S3 menanggapi dengan pengalihan sementara yang memberi tahu pemohon untuk mengirim ulang permintaan ke titik akhir baru. Jika salah membentuk permintaan, Amazon S3 menggunakan pengalihan permanen untuk memberikan arahan tentang cara menjalankan permintaan dengan benar.

Important

Untuk menggunakan fitur ini, Anda harus memiliki aplikasi yang dapat menangani respons pengalihan Amazon S3. Satu-satunya pengecualian adalah untuk aplikasi yang bekerja secara eksklusif dengan bucket yang dibuat tanpa `<CreateBucketConfiguration>`. Untuk informasi selengkapnya tentang batasan lokasi, lihat [Mengakses dan mencantumkan bucket Amazon S3](#).

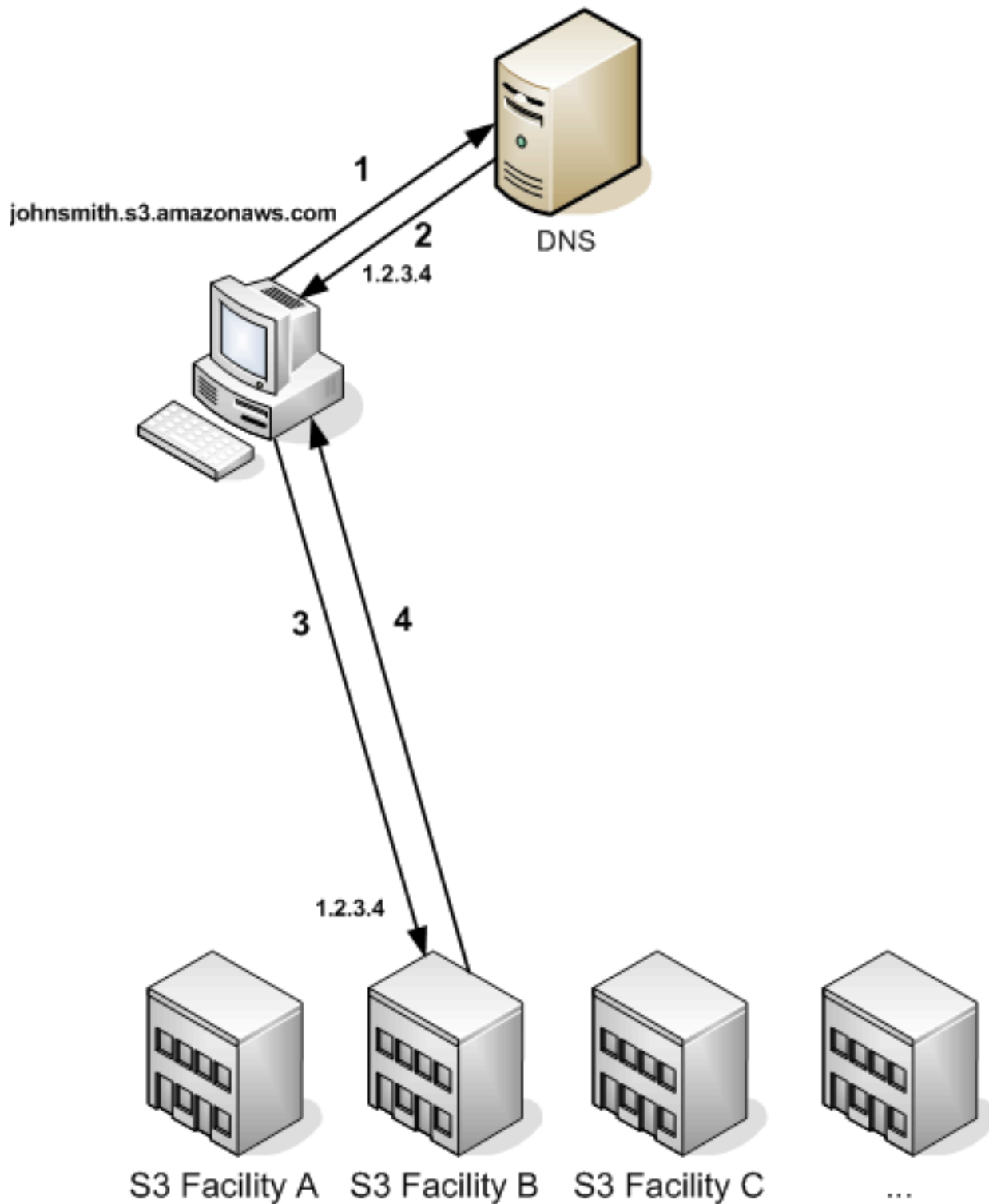
Untuk semua Wilayah yang diluncurkan setelah 20 Maret 2019, jika permintaan tiba di lokasi Amazon S3 yang salah, Amazon S3 mengembalikan kesalahan HTTP 400 Bad Request. Untuk informasi selengkapnya tentang mengaktifkan atau menonaktifkan Wilayah AWS, lihat [Wilayah AWS dan Titik Akhir](#) di Referensi Umum AWS.

Topik

- [Perutean DNS](#)
- [Pengalihan permintaan sementara](#)
- [Pengalihan permintaan permanen](#)
- [Meminta contoh pengalihan](#)

Perutean DNS

Rute perutean DNS meminta fasilitas Amazon S3 yang sesuai. Gambar dan prosedur berikut menunjukkan contoh perutean DNS.



Langkah permintaan perutean DNS

1. Klien mengajukan permintaan DNS untuk mendapatkan objek yang disimpan di Amazon S3.

2. Klien menerima satu atau lebih alamat IP untuk fasilitas yang dapat memproses permintaan. Dalam contoh ini, alamat IP adalah untuk Fasilitas B.
3. Klien membuat permintaan ke Amazon S3 Facility B.
4. Fasilitas B mengembalikan salinan objek kepada klien.

Pengalihan permintaan sementara

Pengalihan sementara adalah jenis respons kesalahan yang memberi sinyal kepada pemohon bahwa mereka harus mengirim ulang permintaan ke titik akhir yang berbeda. Karena sifat Amazon S3, permintaan dapat dikirimkan sementara ke fasilitas yang salah. Ini kemungkinan besar terjadi segera setelah bucket dibuat atau dihapus.

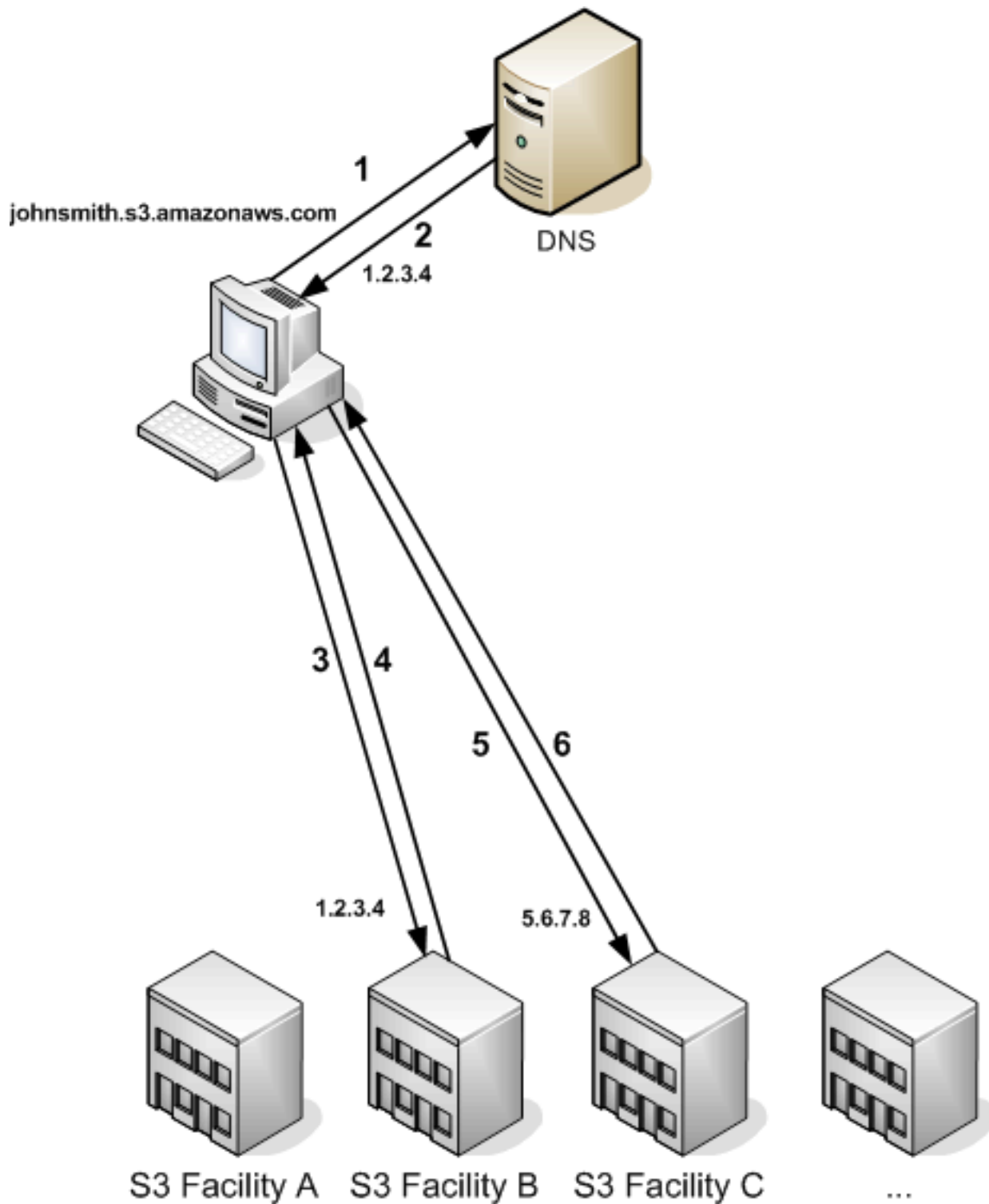
Misalnya, jika Anda membuat bucket baru dan segera membuat permintaan ke bucket, Anda mungkin menerima pengalihan sementara, tergantung pada kendala lokasi bucket. Jika Anda membuat bucket di AS Timur (Virginia Utara) Wilayah AWS, Anda tidak akan melihat pengalihan karena ini juga merupakan titik akhir Amazon S3 default.

Namun, jika bucket dibuat di Wilayah lain, setiap permintaan untuk bucket akan masuk ke titik akhir default sementara entri DNS bucket dipropagasi. Titik akhir default mengarahkan kembali permintaan ke titik akhir yang benar dengan respons HTTP 302. Pengalihan sementara berisi URI ke fasilitas yang benar, yang dapat Anda gunakan untuk segera mengirim ulang permintaan.

Important

Jangan gunakan kembali titik akhir yang disediakan oleh respons pengalihan sebelumnya. Mungkin awalnya tampaknya berhasil (bahkan untuk jangka waktu yang lama), tetapi dapat memberikan hasil yang tidak dapat diprediksi dan pada akhirnya akan gagal tanpa pemberitahuan.

Gambar dan prosedur berikut menunjukkan contoh pengalihan sementara.



Langkah pengalihan permintaan sementara

1. Klien mengajukan permintaan DNS untuk mendapatkan objek yang disimpan di Amazon S3.
2. Klien menerima satu atau lebih alamat IP untuk fasilitas yang dapat memproses permintaan.

3. Klien membuat permintaan ke Amazon S3 Facility B.
4. Fasilitas B mengembalikan pengalihan yang menunjukkan objek tersedia dari Lokasi C.
5. Klien mengirim ulang permintaan ke Fasilitas C.
6. Fasilitas C mengembalikan salinan objek.

Pengalihan permintaan permanen

Pengalihan permanen menunjukkan bahwa permintaan Anda menangani sumber daya secara tidak tepat. Misalnya, pengalihan permanen terjadi jika Anda menggunakan permintaan gaya jalur untuk mengakses bucket yang dibuat menggunakan `<CreateBucketConfiguration>`. Untuk informasi selengkapnya, lihat [Mengakses dan mencantumkan bucket Amazon S3](#).

Untuk membantu Anda menemukan kesalahan ini selama pengembangan, jenis pengalihan ini tidak berisi header HTTP Lokasi yang memungkinkan Anda secara otomatis mengikuti permintaan ke lokasi yang benar. Lihat dokumen kesalahan XML yang dihasilkan untuk bantuan menggunakan titik akhir Amazon S3.

Meminta contoh pengalihan

Berikut ini adalah contoh tanggapan pengalihan permintaan sementara.

API REST sementara mengalihkan respons

```
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>
```

SOAP API sementara mengalihkan respons

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau 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>
```

Pertimbangan DNS

Salah satu persyaratan desain Amazon S3 adalah ketersediaan yang sangat tinggi. Salah satu cara kami memenuhi persyaratan ini adalah dengan memperbarui alamat IP yang terkait dengan titik akhir Amazon S3 di DNS sesuai kebutuhan. Perubahan ini secara otomatis tercermin pada klien berumur pendek, tetapi tidak pada beberapa klien berumur panjang. Long-lived klien perlu mengambil tindakan khusus untuk menyelesaikan kembali titik akhir Amazon S3 secara berkala untuk mendapatkan manfaat dari perubahan ini. Untuk informasi selengkapnya tentang mesin virtual (VM), lihat yang berikut ini:

- Untuk Java, JVM Sun menyimpan pencarian DNS selamanya secara default; buka bagian "InetAddressCaching" dari [InetAddress dokumentasi](#) untuk informasi tentang cara mengubah perilaku ini.
- Untuk PHP, VM PHP persisten yang berjalan di konfigurasi deployment paling populer menyimpan cache pencarian DNS hingga VM dimulai ulang. Pergi [ke dapatkan dokumen HostByName PHP](#).

Mengembangkan dengan Amazon S3 menggunakan AWS CLI

Perintah Amazon S3 AWS CLI diatur ke dalam kumpulan yang berbeda di AWS CLI Referensi Perintah dan masing-masing memiliki perintah yang tersedia sendiri. Jika Anda tidak menemukan perintah yang Anda cari dalam satu set, periksa salah satu set lainnya. Set yang berbeda adalah sebagai berikut:

- `s3`— Menjelaskan perintah tingkat tinggi untuk bekerja dengan objek dan bucket. Ini termasuk tindakan salin, daftar, dan pindahkan. Untuk daftar lengkap perintah dalam set ini, lihat [s3](#) di Referensi AWS CLI Perintah.
- `s3api`— Menjelaskan perintah tingkat rendah yang mengelola sumber daya S3 seperti bucket, objek, sesi, dan unggahan multipart. Perintah ini sesuai dengan [operasi](#) Amazon S3 API. Untuk daftar lengkap perintah CLI dalam set ini, lihat [s3api](#) di Referensi Per AWS CLI intah.
- `s3control`— Menjelaskan perintah tingkat rendah yang mengelola semua sumber daya S3 lainnya seperti Access Grants, grup Storage Lens, dan Amazon S3 di bucket Outposts. Perintah ini sesuai dengan operasi [Amazon S3 Control](#) API. Untuk daftar lengkap perintah CLI dalam set ini, lihat [s3control](#) di Referensi Per AWS CLI intah.

Jika Anda mengalami masalah saat menyiapkan AWS CLI, lihat [Bagaimana cara mengatasi kesalahan “Tidak dapat menemukan kredenSIAL” di Amazon S3](#) di Pusat Peng AWS re:Post etahuan.

Note

Layanan di AWS, seperti Amazon S3, mengharuskan Anda memberikan kredenSIAL saat mengaksesnya. Layanan tersebut kemudian menentukan apakah Anda memiliki izin untuk mengakses sumber daya yang dimiliki oleh layanan. Konsol tersebut memerlukan kata sandi. Anda dapat membuat kunci akses Akun AWS untuk mengakses API AWS CLI atau. Namun, jangan mengakses AWS menggunakan kredenSIAL untuk Anda Akun AWS. Sebagai gantinya, kami sarankan Anda menggunakan AWS Identity and Access Management (IAM). Buat pengguna IAM, tambahkan pengguna tersebut ke grup IAM dengan izin administratif, lalu berikan izin administratif ke pengguna IAM yang sudah dibuat. Kemudian, Anda dapat mengakses AWS dengan menggunakan URL khusus dan kredensial pengguna IAM tersebut. Untuk instruksinya, buka [Membuat Grup Administrator dan pengguna IAM Pertama Anda](#) di Panduan Pengguna IAM.

Pelajari lebih lanjut tentang AWS CLI

Untuk mempelajari lebih lanjut tentang AWS, lihat sumber daya berikut:

- [AWS Command Line Interface](#)
- [AWS Command Line Interface Panduan Pengguna untuk Versi 2](#)

Mengembangkan dengan Amazon S3 menggunakan AWS SDK

AWS perangkat pengembangan perangkat lunak (SDK) tersedia untuk banyak bahasa pemrograman populer. Setiap SDK menyediakan API, contoh kode, dan dokumentasi yang memudahkan developer untuk membangun aplikasi dalam bahasa pilihan mereka.

Note

Anda dapat menggunakan AWS Amplify untuk pengembangan fullstack end-to-end dari aplikasi web dan seluler. Amplify Storage dengan mulus mengintegrasikan kemampuan penyimpanan dan manajemen file ke dalam aplikasi web dan seluler frontend, yang dibangun di atas Amazon S3. Untuk informasi selengkapnya, lihat [Penyimpanan](#) di panduan pengguna Amplify.

Dokumentasi SDK	Contoh kode
AWS SDK untuk C++	AWS SDK untuk C++ contoh kode
AWS CLI	AWS CLI contoh kode
AWS SDK untuk Go	AWS SDK untuk Go contoh kode
AWS SDK untuk Java	AWS SDK untuk Java contoh kode
AWS SDK untuk JavaScript	AWS SDK untuk JavaScript contoh kode
AWS SDK untuk Kotlin	AWS SDK untuk Kotlin contoh kode
AWS SDK untuk .NET	AWS SDK untuk .NET contoh kode

Dokumentasi SDK	Contoh kode
AWS SDK untuk PHP	AWS SDK untuk PHP contoh kode
Alat AWS untuk PowerShell	Alat untuk contoh PowerShell kode
AWS SDK untuk Python (Boto3)	AWS SDK untuk Python (Boto3) contoh kode
AWS SDK untuk Ruby	AWS SDK untuk Ruby contoh kode
AWS SDK for Rust	AWS SDK for Rust contoh kode
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP contoh kode
AWS SDK for Swift	AWS SDK for Swift contoh kode

Untuk contoh spesifik, lihat [Contoh kode untuk Amazon S3 menggunakan AWS SDK](#).

Antarmuka Pemrograman SDK

Setiap AWS SDK menyediakan satu atau lebih antarmuka terprogram untuk bekerja dengan Amazon S3. Setiap SDK menyediakan antarmuka tingkat rendah untuk Amazon S3, dengan metode yang sangat mirip dengan operasi API. Beberapa SDK menyediakan antarmuka tingkat tinggi untuk Amazon S3, yang merupakan abstraksi yang dimaksudkan untuk menyederhanakan kasus penggunaan umum.

Misalnya, saat Anda melakukan upload multipart dengan menggunakan operasi API tingkat rendah, Anda menggunakan operasi untuk memulai upload, operasi lain untuk mengunggah bagian, dan operasi akhir untuk menyelesaikan upload. Operasi API upload multipart tingkat tinggi memungkinkan Anda melakukan semua operasi yang diperlukan untuk mengunggah dalam satu panggilan API. Sebagai contoh, lihat Meng [unggah objek menggunakan unggahan multibagian](#) di Panduan Pengguna Amazon S3.

Low-level Operasi API memungkinkan kontrol yang lebih besar atas unggahan. Sebaiknya gunakan operasi API tingkat rendah jika Anda perlu menjeda dan melanjutkan unggahan, memvariasikan ukuran bagian selama upload, atau memulai upload jika Anda tidak mengetahui ukuran data sebelumnya.

Menentukan Versi Tanda Tangan dalam otentikasi permintaan

Amazon S3 hanya mendukung AWS Signature Version 4 di sebagian besar Wilayah AWS. Di beberapa yang lebih lama Wilayah AWS, Amazon S3 mendukung Signature Version 4 dan Signature Version 2. Akan tetapi, Signature Version 2 sedang dinonaktifkan (dihentikan). Untuk informasi lebih lanjut tentang akhir dukungan untuk Signature Version 2, lihat [AWS Signature Versi 2 dimatikan \(tidak digunakan lagi\) untuk Amazon S3](#).

Untuk daftar dari semua Wilayah Amazon S3 dan versi tanda tangan yang didukung, lihat [Wilayah dan Titik Akhir](#) dalam Referensi Umum AWS .

Untuk semua Wilayah AWS, AWS SDK menggunakan Signature Version 4 secara default untuk mengotentikasi permintaan. Saat menggunakan AWS SDK yang dirilis sebelum Mei 2016, Anda mungkin diminta untuk meminta Tanda Tangan Versi 4, seperti yang ditunjukkan pada tabel berikut.

SDK	Meminta Signature Version 4 untuk Autentikasi Permintaan
AWS CLI	Untuk profil default, jalankan perintah berikut: <pre>\$ aws configure set default.s3.signature_version s3v4</pre> Untuk profil kustom, jalankan perintah berikut: <pre>\$ aws configure set profile.your_profile_name.s3.signature_version s3v4</pre>
SDK Java	Tambahkan hal berikut di kode Anda: <pre>System.setProperty(SDKGlobalConfiguration.ENABLE_S3_SIGV4_SYSTEM_PROPERTY, "true");</pre> Atau, pada baris perintah, tentukan hal berikut: <pre>-Dcom.amazonaws.services.s3.enableV4</pre>
SDK PHP	Tetapkan parameter <code>signature</code> ke v4 saat mengonstruksi klien layanan Amazon S3 untuk PHP SDK v2:

SDK	Meminta Signature Version 4 untuk Autentikasi Permintaan <pre><?php \$client = S3Client::factory(['region' => 'YOUR-REGION', 'version' => 'latest', 'signature' => 'v4']);</pre> Saat menggunakan PHP SDK v3, atur parameter <code>signature_version</code> ke v4 selama konstruksi klien layanan Amazon S3: <pre><?php \$s3 = new Aws\S3\S3Client(['version' => '2006-03-01', 'region' => 'YOUR-REGION', 'signature_version' => 'v4']);</pre>
Python-Boto SDK	Tentukan hal berikut dalam berkas konfigurasi default boto: <pre>[s3] use-sigv4 = True</pre>
Ruby SDK	Ruby SDK-Versi 1: Atur parameter <code>:s3_signature_version</code> ke <code>:v4</code> saat mengonstruksi klien: <pre>s3 = AWS::S3::Client.new(:s3_signature_version => :v4)</pre> Ruby SDK-Versi 3: Atur parameter <code>signature_version</code> ke v4 saat mengonstruksi klien: <pre>s3 = Aws::S3::Client.new(signature_version: 'v4')</pre>

SDK	Meminta Signature Version 4 untuk Autentikasi Permintaan
SDK .NET	Tambahkan yang berikut ini ke kode sebelum membuat klien Amazon S3: <pre>AWSConfigsS3.UseSignatureVersion4 = true;</pre> Atau, tambahkan yang berikut ini ke file konfigurasi: <pre><appSettings> <add key="AWS.S3.UseSignatureVersion4" value="true" /> </appSettings></pre>

AWS Signature Versi 2 dimatikan (tidak digunakan lagi) untuk Amazon S3

Signature Version 2 dinonaktifkan (dihentikan) di Amazon S3. Amazon S3 kemudian hanya akan menerima permintaan API yang ditandatangani menggunakan Signature Version 4.

Bagian ini memberikan jawaban atas pertanyaan umum mengenai akhir dukungan untuk Signature Version 2.

Apa itu Versi Tanda 2/4 Tangan, dan Apa Artinya Menandatangani Permintaan?

Proses penandatanganan Signature Version 2 atau Signature Version 4 digunakan untuk mengautentikasi permintaan Amazon S3 API Anda. Permintaan penandatanganan memungkinkan Amazon S3 mengidentifikasi siapa yang mengirim permintaan dan melindungi permintaan Anda dari pelaku kejahatan.

Untuk informasi selengkapnya tentang AWS permintaan penandatangan, [lihat Menandatangani Permintaan AWS API](#) di Referensi Umum AWS.

Pembaruan Apa yang Anda Buat?

Saat ini kami mendukung permintaan API Amazon S3 yang ditandatangani menggunakan proses Signature Version 2 dan Signature Version 4. Setelah itu, Amazon S3 kemudian hanya akan menerima permintaan yang ditandatangani menggunakan Signature Version 4.

Untuk informasi selengkapnya tentang AWS permintaan penandatanganan, lihat [Perubahan Tanda Tangan Versi 4](#) di Referensi Umum AWS.

Mengapa Anda Membuat Pembaruan?

Signature Version 4 memberikan keamanan yang lebih baik dengan menggunakan kunci penandatanganan dan bukan kunci akses rahasia Anda. Signature Version 4 saat ini didukung di semua Wilayah AWS, sedangkan Signature Version 2 hanya didukung di Wilayah yang diluncurkan sebelum Januari 2014. Pembaruan ini memungkinkan kami memberikan pengalaman yang lebih konsisten di seluruh Wilayah.

Bagaimana Saya Memastikan bahwa Saya Menggunakan Signature Version 4, dan Pembaruan Apa yang Saya Perlukan?

Versi tanda tangan yang digunakan untuk menandatangani permintaan Anda biasanya ditetapkan oleh alat atau SDK di sisi klien. Secara default, versi terbaru AWS SDK kami menggunakan Signature Version 4. Untuk perangkat lunak pihak ketiga, hubungi tim dukungan yang sesuai untuk perangkat lunak Anda guna mengonfirmasi versi apa yang Anda butuhkan. Jika Anda mengirim panggilan REST langsung ke Amazon S3, Anda harus memodifikasi aplikasi Anda untuk menggunakan proses penandatanganan Signature Version 4.

Untuk informasi tentang versi SDK mana yang AWS akan digunakan saat pindah ke Signature Version 4, lihat [Beralih dari Signature Version 2 ke Signature Version 4](#).

Untuk informasi tentang menggunakan Signature Version 4 dengan API REST Amazon S3, lihat [Permintaan Autentikasi \(Signature Version 4 AWS\)](#) di Referensi API Amazon Simple Storage Service.

Apa yang Terjadi jika Saya Tidak Melakukan Pembaruan?

Permintaan yang ditandatangani dengan Signature Version 2 yang dibuat setelahnya akan gagal diautentikasi dengan Amazon S3. Pemohon akan melihat kesalahan yang menyatakan bahwa permintaan harus ditandatangani dengan Signature Version 4.

Haruskah Saya Melakukan Perubahan Meskipun Saya Menggunakan URL yang Ditandatangani Sebelumnya yang Harus Saya Tandatangani Selama Lebih dari 7 Hari?

Jika Anda menggunakan URL yang ditandatangani sebelumnya yang mengharuskan Anda menandatangani lebih dari 7 hari, saat ini tidak ada tindakan yang diperlukan. Anda dapat terus

menggunakan Tanda Tangan Versi AWS 2 untuk menandatangani dan mengotentikasi URL yang telah ditandatangani sebelumnya. Untuk detail selengkapnya tentang cara bermigrasi ke Tanda Tangan Versi 4 untuk skenario URL yang telah ditandatangani sebelumnya, lihat dokumentasi Amazon S3.

Info Selengkapnya

- Untuk informasi selengkapnya tentang menggunakan Tanda Tangan Versi 4, lihat [Menandatangani AWS Permintaan API](#).
- Lihat daftar perubahan antara Signature Version 2 dan Signature Version 4 di [Perubahan dalam Signature Version 4](#).
- Lihat posting Tanda [AWS Tanda Tangan Versi 4 untuk menggantikan AWS Tanda Tangan Versi 2 untuk menandatangani permintaan Amazon S3 API](#) di AWS forum.
- Jika Anda memiliki pertanyaan atau kekhawatiran, hubungi [Dukungan](#).

Beralih dari Signature Version 2 ke Signature Version 4

Jika saat ini Anda menggunakan Signature Version 2 untuk autentikasi permintaan API Amazon S3, Anda harus beralih ke menggunakan Signature Version 4. Dukungan berakhir untuk Signature Version 2, sebagaimana dijelaskan di [AWS Signature Versi 2 dimatikan \(tidak digunakan lagi\) untuk Amazon S3](#).

Untuk informasi tentang menggunakan Signature Version 4 dengan API REST Amazon S3, lihat [Permintaan Autentikasi \(Signature Version 4 AWS\)](#) di Referensi API Amazon Simple Storage Service.

Tabel berikut mencantumkan SDK dengan versi minimum yang diperlukan untuk menggunakan Signature Version 4 (SigV4). Jika Anda menggunakan URL yang telah ditandatangani sebelumnya dengan AWS SDK Java, JavaScript (Node.js), atau Python (Boto/CLI), Anda harus mengatur yang benar Wilayah AWS dan mengatur Signature Version 4 dalam konfigurasi klien. Untuk informasi tentang pengaturan SigV4 di konfigurasi klien, lihat [Menentukan Versi Tanda Tangan dalam otentikasi permintaan](#).

Jika Anda menggunakan ini SDK/Product	Tingkatkan ke versi SDK ini	Perubahan kode yang diperlukan bagi klien untuk menggunakan Sigv4?	Tautan ke dokumentasi SDK
AWS SDK untuk Java v1	Peningkatan ke Java 1.11.201+ atau v2.	Ya	Menentukan Versi Tanda Tangan dalam otentikasi permintaan
AWS SDK untuk Java v2	Tidak diperlukan peningkatan SDK.	Tidak	AWS SDK untuk Java
AWS SDK untuk .NET v1	Tingkatkan ke 3.1.10 atau lebih baru.	Ya	AWS SDK untuk .NET
AWS SDK untuk .NET v2	Tingkatkan ke 3.1.10 atau lebih baru.	Tidak	AWS SDK untuk .NET v2
AWS SDK untuk .NET v3	Tingkatkan ke 3.3.0.0 atau lebih baru.	Ya	AWS SDK untuk .NET v3
AWS SDK untuk JavaScript v1	Tingkatkan ke 2.68.0 atau lebih baru.	Ya	AWS SDK untuk JavaScript

Jika Anda menggunakan ini SDK/Product	Tingkatkan ke versi SDK ini	Perubahan kode yang diperlukan bagi klien untuk menggunakan Sigv4?	Tautan ke dokumentasi SDK
AWS SDK untuk JavaScript v2	Tingkatkan ke 2.68.0 atau lebih baru.	Ya	AWS SDK untuk JavaScript
AWS SDK untuk JavaScript v3	Saat ini tidak ada tindakan yang diperlukan. Tingkatkan ke versi utama V3 di Q3 2019.	Tidak	AWS SDK untuk JavaScript
AWS SDK untuk PHP v1	Rekomendasi untuk meningkatkan ke versi terbaru PHP atau, setidaknya ke v2.7.4 dengan parameter tanda tangan yang diatur ke v4 dalam konfigurasi klien S3.	Ya	AWS SDK untuk PHP

Jika Anda menggunakan ini SDK/Product	Tingkatkan ke versi SDK ini	Perubahan kode yang diperlukan bagi klien untuk menggunakan Sigv4?	Tautan ke dokumentasi SDK
AWS SDK untuk PHP v2	Rekomendasikan untuk meningkatkan ke versi terbaru PHP atau, setidaknya ke v2.7.4 dengan parameter tanda tangan yang diatur ke v4 dalam konfigurasi klien S3.	Tidak	AWS SDK untuk PHP
AWS SDK untuk PHP v3	Tidak diperlukan peningkatan SDK.	Tidak	AWS SDK untuk PHP
Boto2	Tingkatkan ke Boto2 v2.49.0.	Ya	Peningkatan Boto 2
Boto3	Tingkatkan ke 1.5.71 (Botocore), 1.4.6 (Boto3).	Ya	Boto 3 - AWS SDK untuk Python

Jika Anda menggunakan ini SDK/Product	Tingkatkan ke versi SDK ini	Perubahan kode yang diperlukan bagi klien untuk menggunakan Sigv4?	Tautan ke dokumentasi SDK
AWS CLI	Memutakhirkan ke 1.11.108.	Ya	AWS Command Line Interface
AWS CLI v2 (pratinjau)	Tidak diperlukan peningkatan SDK.	Tidak	AWS Command Line Interface versi 2
AWS SDK untuk Ruby v1	Tingkatkan ke Ruby V3.	Ya	Ruby V3 untuk AWS
AWS SDK untuk Ruby v2	Tingkatkan ke Ruby V3.	Ya	Ruby V3 untuk AWS
AWS SDK untuk Ruby v3	Tidak diperlukan peningkatan SDK.	Tidak	Ruby V3 untuk AWS
Go	Tidak diperlukan peningkatan SDK.	Tidak	AWS SDK untuk Go

Jika Anda menggunakan ini SDK/Product	Tingkatkan ke versi SDK ini	Perubahan kode yang diperlukan bagi klien untuk menggunakan Sigv4?	Tautan ke dokumentasi SDK
C++	Tidak diperlukan peningkatan SDK.	Tidak	AWS SDK untuk C++

AWS Tools for Windows PowerShell atau AWS Tools for PowerShell Core

Jika Anda menggunakan versi modul lebih awal dari 3.3.0.0, Anda harus meningkatkan ke 3.3.0.0.

Untuk mendapatkan informasi versi, gunakan cmdlet `Get-Module`:

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

Untuk memperbarui versi 3.3.0.0, gunakan cmdlet `Update-Module`:

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

Anda dapat menggunakan URL yang telah ditandatangani sebelumnya yang valid selama lebih dari 7 hari yang akan Anda gunakan untuk mengirimkan lalu lintas Signature Version 2.

Mendapatkan ID permintaan Amazon S3 untuk AWS Dukungan

Setiap kali Anda menghubungi AWS Dukungan karena mengalami kesalahan atau perilaku tak terduga di Amazon S3, Anda harus memberikan ID permintaan yang terkait dengan tindakan yang

gagal. AWS Dukungan menggunakan ID permintaan ini untuk membantu menyelesaikan masalah yang Anda alami.

ID Permintaan hadir berpasangan, dikembalikan dalam setiap respons yang diproses Amazon S3 (bahkan respons yang salah), dan dapat diakses melalui log verbose. Ada sejumlah metode umum untuk mendapatkan ID permintaan Anda, termasuk log akses server S3 dan AWS CloudTrail peristiwa atau peristiwa data.

Setelah Anda memulihkan log ini, salin dan simpan kedua nilai tersebut, karena Anda akan membutuhkannya saat Anda menghubungi AWS Dukungan. Untuk informasi tentang menghubungi AWS Dukungan, lihat [Kon tak AWS](#) atau [AWS Dukungan Dokumentasi](#).

Topik

- [Menggunakan AWS CLI untuk mendapatkan ID permintaan](#)
- [Menggunakan Windows PowerShell untuk mendapatkan ID permintaan](#)
- [Penggunaan AWS CloudTrail peristiwa data untuk mendapatkan ID permintaan](#)
- [Menggunakan pencatatan log akses server S3 untuk mendapatkan ID permintaan](#)
- [Menggunakan HTTP untuk memperoleh ID Permintaan](#)
- [Menggunakan peramban web untuk memperoleh ID permintaan](#)
- [Menggunakan AWS SDK untuk mendapatkan ID permintaan](#)

Menggunakan AWS CLI untuk mendapatkan ID permintaan

Untuk mendapatkan ID permintaan Anda saat menggunakan AWS Command Line Interface (AWS CLI), tambahkan `--debug` ke perintah Anda. Misalnya Anda harus melihat `x-amz-request-id` dan `x-amz-id-2` di log debug seperti yang ditunjukkan di bawah ini:

```
...
2025-04-30 14:35:28,572 - MainThread - botocore.parsers - DEBUG - Response headers:
{'x-amz-id-2': 'a+zm50vDJH3LLmCiXvwEo0u0PtPS/qCJaBvB2ZMH9dzyzTiJhiLZkBFFoRfsPf0KztUKT/
garCI=', 'x-amz-request-id': 'N4NMN0MJ4VDFZMX9', 'Date': 'Wed, 30 Apr 2025 21:35:29
GMT', 'Content-Type': 'application/xml', 'Transfer-Encoding': 'chunked', 'Server':
'AmazonS3'}
...
```

Menggunakan Windows PowerShell untuk mendapatkan ID permintaan

Untuk informasi tentang memulihkan log dengan Windows PowerShell, lihat Login [Respon AWS Tools for Windows PowerShell](#) di Blog Alat AWS Pengembang.

Penggunaan AWS CloudTrail peristiwa data untuk mendapatkan ID permintaan

Bucket Amazon S3 yang dikonfigurasi dengan peristiwa CloudTrail data untuk mencatat operasi API tingkat objek S3 menyediakan informasi terperinci tentang tindakan yang diambil oleh pengguna, peran, atau AWS layanan di Amazon S3. Anda dapat mengidentifikasi ID permintaan S3 dengan menanyakan CloudTrail peristiwa dengan Athena.

Untuk informasi selengkapnya, lihat [Mengidentifikasi permintaan Amazon S3 menggunakan CloudTrail](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Menggunakan pencatatan log akses server S3 untuk mendapatkan ID permintaan

Bucket Amazon S3 yang dikonfigurasi untuk pencatatan log akses server Amazon S3 memberikan catatan yang detail untuk setiap permintaan yang dibuat ke bucket. Anda dapat mengidentifikasi ID permintaan S3 dengan melakukan kueri log akses server menggunakan Athena.

Untuk informasi selengkapnya, lihat [Mengkueri log akses untuk permintaan dengan menggunakan Amazon Athena](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Menggunakan HTTP untuk memperoleh ID Permintaan

Anda dapat memperoleh ID permintaan Anda, `x-amz-request-id` dan `x-amz-id-2` dengan mencatat bit dari permintaan HTTP sebelum itu mencapai aplikasi target. Ada berbagai alat pihak ketiga yang dapat digunakan untuk memulihkan log verbose untuk permintaan HTTP. Pilih satu yang Anda percaya, lalu jalankan alat untuk mendengarkan di port yang dilalui lalu lintas Amazon S3 Anda, saat Anda mengirim permintaan HTTP Amazon S3 lainnya.

Untuk permintaan HTTP, sepasang ID permintaan akan terlihat seperti berikut:

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

 Note

Permintaan HTTPS dienkripsi dan tersembunyi di sebagian besar tangkapan paket.

Menggunakan peramban web untuk memperoleh ID permintaan

Sebagian besar peramban web memiliki alat pengembang yang dapat Anda gunakan untuk melihat header permintaan.

Untuk permintaan berbasis peramban web yang memunculkan kesalahan, pasangan ID permintaan akan terlihat seperti contoh berikut.

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

Untuk mendapatkan pasangan ID permintaan dari permintaan yang berhasil, gunakan alat pengembang peramban Anda untuk melihat header respons HTTP.

Menggunakan AWS SDK untuk mendapatkan ID permintaan

Bagian berikut mencakup informasi untuk mengonfigurasi logging dengan menggunakan AWS SDK yang berbeda.

 Note

Meskipun Anda dapat mengaktifkan logging bertele-tele pada setiap permintaan dan respons, jangan sarankan mengaktifkan logging dalam sistem produksi, karena permintaan atau tanggapan yang besar dapat memperlambat aplikasi secara signifikan.

Untuk permintaan AWS SDK, pasangan ID permintaan akan terlihat seperti berikut:

```
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++

Jenis logger dan verbositas ditentukan selama inisialisasi SDK dalam argumen. `SDKOptions`

Contoh berikut menentukan tingkat verbositas sebagai. `LogLevel::Debug`

Logger default akan menulis ke sistem file dan file diberi nama menggunakan konvensi `aws_sdk_YYYY-MM-DD-HH.log` di bawah ini. Logger membuat file baru pada jam tersebut.

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

Untuk informasi selengkapnya, lihat [Bagaimana cara mengaktifkan logging?](#) di AWS SDK untuk wiki C++ pada GitHub.

Go

Anda dapat mengonfigurasi logging dengan menggunakan SDK for Go. Untuk informasi selengkapnya, [lihat Log](#) in di Panduan Pengembang AWS SDK untuk Go v2.

Java

Anda dapat mengaktifkan logging untuk permintaan atau tanggapan tertentu untuk menangkap dan mengembalikan hanya header yang relevan. Untuk melakukannya, impor kelas `com.amazonaws.services.s3.S3ResponseMetadata`. Setelah itu, Anda dapat menyimpan permintaan dalam variabel sebelum melakukan permintaan aktual. Untuk mendapatkan permintaan atau respons yang dicatat, panggil `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());
```

Atau, Anda dapat menggunakan pencatatan log verbose untuk setiap permintaan dan respons Java. Untuk informasi selengkapnya, lihat [Verbose Wire Logging](#) di Panduan Pengembang AWS SDK untuk Java .

JavaScript

Ini AWS SDK untuk JavaScript memiliki logger bawaan sehingga Anda dapat mencatat panggilan API yang Anda buat dengannya. Untuk mengaktifkan logger dan mencetak entri log di konsol, tambahkan pernyataan berikut ke kode Anda:

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

Untuk informasi selengkapnya, lihat [Pencat AWS SDK untuk JavaScript](#) atau Panggilan di Panduan AWS SDK untuk JavaScript Pengembang.

Kotlin

Dengan AWS SDK untuk Kotlin, Anda dapat menentukan mode log untuk pesan tingkat kabel menggunakan kode atau pengaturan lingkungan. Anda dapat mengatur mode log untuk permintaan HTTP dan tanggapan HTTP.

Untuk memilih logging tambahan, atur `logMode` properti saat Anda membuat klien layanan:

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

// ...

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

Atau, Anda dapat mengatur mode log menggunakan variabel lingkungan:

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

Untuk informasi selengkapnya, lihat [Log](#) in di Panduan AWS SDK untuk Kotlin Pengembang.

.NET

Anda dapat mengonfigurasi logging SDK untuk .NET dengan menggunakan alat `System.Diagnostics` logging bawaan. Untuk informasi selengkapnya, lihat posting [Logging dengan Blog SDK untuk .NET](#) AWS Pengembang.

Note

Secara default, log yang dikembalikan hanya berisi informasi kesalahan. Untuk mendapatkan ID permintaan, file konfigurasi harus memiliki `AWSLogMetrics` (dan secara opsional, `AWSResponseLogging`) ditambahkan.

PHP

Anda bisa mendapatkan informasi debug, termasuk data yang dikirim melalui kabel, dengan mengatur opsi debug ke `true` dalam konstruktor klien.

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

Untuk informasi lebih lanjut, lihat [Bagaimana cara melihat data apa yang dikirim melalui kabel?](#) di Panduan Pengembang AWS SDK untuk PHP .

Python (Boto3)

Dengan AWS SDK untuk Python (Boto3), Anda dapat mencatat tanggapan tertentu. Anda dapat menggunakan fitur ini agar hanya menangkap header yang relevan. Kode berikut menunjukkan cara mencatat bagian dari respons terhadap 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'])
```

Anda juga dapat menangkap pengecualian dan mencatat informasi yang relevan ketika pengecualian diajukan. Untuk informasi selengkapnya, lihat [Membedakan informasi berguna dari respons kesalahan](#) di Referensi API SDK untuk Python AWS (Boto).

Selain itu, Anda dapat mengonfigurasi Boto3 agar mengeluarkan log debug verbose menggunakan kode berikut:

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

Untuk informasi selengkapnya, lihat [set_stream_logger](#) di Referensi API SDK untuk Python AWS (Boto).

Ruby

Anda bisa mendapatkan ID permintaan Anda menggunakan SDK untuk Ruby Versi 3. Anda dapat mengaktifkan pencatatan kawat HTTP di klien Anda menggunakan kode berikut:

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

Note

Wire logging dapat mencakup informasi sensitif, seperti ID kunci akses Anda. Informasi sensitif harus dibersihkan sebelum dibagikan dengan AWS Dukungan.

Anda juga dapat menemukan ID permintaan objek konteks permintaan dalam respons permintaan atau kesalahan:

```
# 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]
```

Untuk informasi selengkapnya, lihat [Men-debug menggunakan informasi pelacakan AWS SDK untuk Ruby kabel dari klien](#) di Panduan AWS SDK untuk Ruby Pengembang.

Rust

Untuk mengaktifkan logging, tambahkan tracing-subscriber peti dan inisialisasi di aplikasi Rust Anda.

Tambahkan pustaka penelusuran ke Cargo.toml file Anda:

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

Kemudian, dalam kode Rust Anda, inisialisasi logger di fungsi utama sebelum Anda memanggil operasi SDK apa pun:

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

Untuk informasi selengkapnya, [lihat Mengonfigurasi dan menggunakan AWS SDK for Rust](#) log di Panduan AWS SDK for Rust Pengembang.

Operasi API tingkat objek Amazon S3 yang didukung untuk Tabel S3

Tabel berikut mencakup operasi API tingkat objek S3 yang didukung dan header yang sesuai untuk Tabel S3. Untuk daftar API Tabel Amazon S3, lihat Tabel [Amazon S3](#). Untuk informasi selengkapnya tentang Tabel Amazon S3, lihat Bek [erja dengan Tabel Amazon S3 dan bucket tabel](#) di Panduan Pengguna Amazon S3.

Object-level API	Header yang didukung	Catatan	
AbortMultiupload	x-amz-expected-bucket-owner	Tidak ada	
CompleteMultiupload	x-amz-checksum-crc32	Tidak ada	
	x-amz-checksum-crc32c	Tidak ada	

Object-level API	Header yang didukung	Catatan	
	x-amz-checksum-sha1	Tidak ada	
	x-amz-checksum-sha256	Tidak ada	
	x-amz-expected-bucket-owner	Tidak ada	
	If-Match	Tidak ada	
	If-None-Match	Tidak ada	
CreateMultipartUpload	x-amz-acl: ACL	Untuk Tabel S3, nilai default bucket-owner-full-control nya adalah dan tidak dapat diubah.	
	Cache-Control	Tidak ada	
	Content-Disposition	Tidak ada	
	Content-Encoding	Tidak ada	
	Content-Language	Tidak ada	
	Content-Type	Tidak ada	
	Expires	Tidak ada	
	x-amz-checksum-algorithm	Tidak ada	

Object-level API	Header yang didukung	Catatan	
	x-amz-storage-class	Untuk Tabel S3, nilai default STANDARD nya adalah dan tidak dapat diubah.	
	x-amz-server-side-encryption	Untuk Tabel S3, nilai defaultnya adalah (SSE-S3AES256) () dan tidak dapat diubah.	
GetObject	If-Match	Tidak ada	
	If-Modified-Since	Tidak ada	
	If-None-Match	Tidak ada	
	If-Unmodified-Since	Tidak ada	
	Range	Tidak ada	
	x-amz-expected-bucket-owner	Tidak ada	
	x-amz-checksum-mode	Tidak ada	
HeadObject	If-Match	Tidak ada	
	If-Modified-Since	Tidak ada	
	If-None-Match	Tidak ada	
	If-Unmodified-Since	Tidak ada	
	Range	Tidak ada	

Object-level API	Header yang didukung	Catatan	
	x-amz-checksum-mode	Tidak ada	
	x-amz-expected-bucket-owner	Tidak ada	
ListParts	x-amz-expected-bucket-owner	Tidak ada	
PutObject	x-amz-acl: ACL	Untuk Tabel S3, nilai default bucket-owner-full-control nya adalah dan tidak dapat diubah.	
	Cache-Control	Tidak ada	
	Content-Disposition	Tidak ada	
	Content-Encoding	Tidak ada	
	Content-Language	Tidak ada	
	Content-Length	Tidak ada	
	Content-MD5	Tidak ada	
	Content-Type	Tidak ada	
	x-amz-sdk-checksum-algorithm	Tidak ada	
	x-amz-checksum-crc32	Tidak ada	

Object-level API	Header yang didukung	Catatan	
	x-amz-checksum-crc32c	Tidak ada	
	x-amz-checksum-sha1	Tidak ada	
	x-amz-checksum-sha256	Tidak ada	
	Expires	Tidak ada	
	If-None-Match	Tidak ada	
	x-amz-expected-bucket-owner	Tidak ada	
	x-amz-storage-class	Untuk Tabel S3, nilai default STANDARD nya adalah dan tidak dapat diubah.	
	x-amz-server-side-encryption	Untuk Tabel S3, nilai defaultnya adalah (SSE-S3AES256) () dan tidak dapat diubah.	
UploadPart	Content-Length	Tidak ada	
	Content-MD5	Tidak ada	
	x-amz-checksum-crc32	Tidak ada	
	x-amz-checksum-crc32c	Tidak ada	

Object-level API	Header yang didukung	Catatan	
	x-amz-checksum-sha1	Tidak ada	
	x-amz-checksum-sha256	Tidak ada	
	x-amz-expected-bucket-owner	Tidak ada	

Contoh kode untuk Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan Amazon S3 dengan kit pengembangan AWS perangkat lunak (SDK).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Contoh kode

- [Contoh kode untuk Amazon S3 menggunakan AWS SDK](#)
 - [Contoh dasar untuk Amazon S3 menggunakan AWS SDK](#)
 - [Halo Amazon S3](#)
 - [Pelajari dasar-dasar Amazon S3 dengan AWS SDK](#)
 - [Tindakan untuk Amazon S3 menggunakan AWS SDK](#)
 - [Gunakan AbortMultipartUploaddengan AWS SDK atau CLI](#)
 - [Gunakan CompleteMultipartUploaddengan AWS SDK atau CLI](#)
 - [Gunakan CopyObjectdengan AWS SDK atau CLI](#)
 - [Gunakan CreateBucketdengan AWS SDK atau CLI](#)
 - [Gunakan CreateMultiRegionAccessPointdengan AWS SDK](#)
 - [Gunakan CreateMultipartUploaddengan AWS SDK atau CLI](#)
 - [Gunakan CreatePresignedPostdengan AWS SDK](#)
 - [Gunakan DeleteBucketdengan AWS SDK atau CLI](#)
 - [Gunakan DeleteBucketAnalyticsConfigurationdengan CLI](#)
 - [Gunakan DeleteBucketCorsdengan AWS SDK atau CLI](#)
 - [Gunakan DeleteBucketEncryptiondengan CLI](#)
 - [Gunakan DeleteBucketInventoryConfigurationdengan CLI](#)
 - [Gunakan DeleteBucketLifecycledengan AWS SDK atau CLI](#)
 - [Gunakan DeleteBucketMetricsConfigurationdengan CLI](#)
 - [Gunakan DeleteBucketPolicydengan AWS SDK atau CLI](#)
 - [Gunakan DeleteBucketReplicationdengan CLI](#)
 - [Gunakan DeleteBucketTaggingdengan CLI](#)

- [Gunakan DeleteBucketWebsitedengan AWS SDK atau CLI](#)
- [Gunakan DeleteObjectdengan AWS SDK atau CLI](#)
- [Gunakan DeleteObjectTaggingdengan CLI](#)
- [Gunakan DeleteObjectsdengan AWS SDK atau CLI](#)
- [Gunakan DeletePublicAccessBlockdengan CLI](#)
- [Gunakan GetBucketAccelerateConfigurationdengan CLI](#)
- [Gunakan GetBucketAcldengan AWS SDK atau CLI](#)
- [Gunakan GetBucketAnalyticsConfigurationdengan CLI](#)
- [Gunakan GetBucketCorsdengan AWS SDK atau CLI](#)
- [Gunakan GetBucketEncryptiondengan AWS SDK atau CLI](#)
- [Gunakan GetBucketInventoryConfigurationdengan CLI](#)
- [Gunakan GetBucketLifecycleConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan GetBucketLocationdengan AWS SDK atau CLI](#)
- [Gunakan GetBucketLoggingdengan CLI](#)
- [Gunakan GetBucketMetricsConfigurationdengan CLI](#)
- [Gunakan GetBucketNotificationdengan CLI](#)
- [Gunakan GetBucketPolicydengan AWS SDK atau CLI](#)
- [Gunakan GetBucketPolicyStatusdengan CLI](#)
- [Gunakan GetBucketReplicationdengan AWS SDK atau CLI](#)
- [Gunakan GetBucketRequestPaymentdengan CLI](#)
- [Gunakan GetBucketTaggingdengan CLI](#)
- [Gunakan GetBucketVersioningdengan CLI](#)
- [Gunakan GetBucketWebsitedengan AWS SDK atau CLI](#)
- [Gunakan GetObjectdengan AWS SDK atau CLI](#)
- [Gunakan GetObjectAcldengan AWS SDK atau CLI](#)
- [Gunakan GetObjectAttributesdengan AWS SDK atau CLI](#)
- [Gunakan GetObjectLegalHolddengan AWS SDK atau CLI](#)
- [Gunakan GetObjectLockConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan GetObjectRetentiondengan AWS SDK atau CLI](#)
- [Gunakan GetObjectTaggingdengan CLI](#)

- [Gunakan GetPublicAccessBlockdengan CLI](#)
- [Gunakan HeadBucketdengan AWS SDK atau CLI](#)
- [Gunakan HeadObjectdengan AWS SDK atau CLI](#)
- [Gunakan ListBucketAnalyticsConfigurationsdengan CLI](#)
- [Gunakan ListBucketInventoryConfigurationsdengan CLI](#)
- [Gunakan ListBucketsdengan AWS SDK atau CLI](#)
- [Gunakan ListMultipartUploadsdengan AWS SDK atau CLI](#)
- [Gunakan ListObjectVersionsdengan AWS SDK atau CLI](#)
- [Gunakan ListObjectsdengan CLI](#)
- [Gunakan ListObjectsV2 dengan AWS SDK atau CLI](#)
- [Gunakan PutBucketAccelerateConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketAcldengan AWS SDK atau CLI](#)
- [Gunakan PutBucketCorsdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketEncryptiondengan AWS SDK atau CLI](#)
- [Gunakan PutBucketLifecycleConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketLoggingdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketNotificationdengan CLI](#)
- [Gunakan PutBucketNotificationConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketPolicydengan AWS SDK atau CLI](#)
- [Gunakan PutBucketReplicationdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketRequestPaymentdengan CLI](#)
- [Gunakan PutBucketTaggingdengan CLI](#)
- [Gunakan PutBucketVersioningdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketWebsitedengan AWS SDK atau CLI](#)
- [Gunakan PutObjectdengan AWS SDK atau CLI](#)
- [Gunakan PutObjectAcldengan AWS SDK atau CLI](#)
- [Gunakan PutObjectLegalHolddengan AWS SDK atau CLI](#)
- [Gunakan PutObjectLockConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan PutObjectRetentiondengan AWS SDK atau CLI](#)
- [Gunakan RestoreObjectdengan AWS SDK atau CLI](#)

- [Gunakan SelectObjectContent dengan AWS SDK atau CLI](#)
- [Gunakan UploadPart dengan AWS SDK atau CLI](#)
- [Gunakan UploadPartCopy dengan AWS SDK atau CLI](#)
- [Skenario untuk Amazon S3 menggunakan AWS SDK](#)
 - [Periksa apakah ada ember](#)
 - [Periksa apakah ada objek](#)
 - [Mengkonversi teks menjadi ucapan dan kembali ke teks menggunakan AWS SDK](#)
 - [Buat URL yang telah ditetapkan sebelumnya untuk Amazon S3 menggunakan AWS SDK](#)
 - [Membuat aplikasi manajemen aset foto yang memungkinkan pengguna mengelola foto menggunakan label](#)
 - [Halaman web yang mencantumkan objek Amazon S3 menggunakan AWS SDK](#)
 - [Membuat aplikasi penjelajah Amazon Textract](#)
 - [Hapus semua objek dalam bucket Amazon S3 tertentu menggunakan AWS SDK](#)
 - [Hapus unggahan multipart yang tidak lengkap ke Amazon S3 menggunakan AWS SDK](#)
 - [Mendeteksi APD dalam gambar dengan Amazon Rekognition menggunakan AWS SDK](#)
 - [Mendeteksi entitas dalam teks yang diekstraksi dari gambar menggunakan AWS SDK](#)
 - [Mendeteksi wajah dalam gambar menggunakan AWS SDK](#)
 - [Mendeteksi objek dalam gambar dengan Amazon Rekognition menggunakan AWS SDK](#)
 - [Mendeteksi orang dan objek dalam video dengan Amazon Rekognition menggunakan AWS SDK](#)
 - [Unduh 'direktori' S3 dari bucket Amazon Simple Storage Service \(Amazon S3\)](#)
 - [Mengunduh semua objek di bucket Amazon Simple Storage Service \(Amazon S3\) ke direktori lokal](#)
 - [Unduh aliran dengan ukuran yang tidak diketahui dari objek Amazon S3 menggunakan AWS SDK](#)
 - [Dapatkan objek Amazon S3 dari Multi-Region Access Point dengan menggunakan AWS SDK](#)
 - [Dapatkan objek dari bucket Amazon S3 menggunakan AWS SDK, menentukan header If-Modified-Since](#)
 - [Memulai enkripsi untuk objek Amazon S3 menggunakan AWS SDK](#)
 - [Memulai tag untuk objek Amazon S3 menggunakan AWS SDK](#)

- [Memulai dengan cluster pemrosesan data besar](#)
- [Memulai dengan manajemen konfigurasi](#)
- [Memulai dengan ekstraksi teks dokumen](#)
- [Memulai dengan toko fitur pembelajaran mesin](#)
- [Memulai dengan penyimpanan objek](#)
- [Memulai dengan analisis kueri](#)
- [Bekerja dengan fitur kunci objek Amazon S3 menggunakan AWS SDK](#)
- [Buat permintaan bersyarat Amazon S3 menggunakan AWS SDK](#)
- [Mengelola daftar kontrol akses \(ACL\) untuk bucket Amazon S3 menggunakan AWS SDK](#)
- [Mengelola pesan Amazon SQS besar menggunakan Amazon S3 dengan AWS SDK](#)
- [Mengelola objek Amazon S3 berversi dalam batch dengan fungsi Lambda menggunakan AWS SDK](#)
- [Mengurai URI Amazon S3 menggunakan AWS SDK](#)
- [Melakukan salinan multipart dari objek Amazon S3 menggunakan AWS SDK](#)
- [Menerima dan memproses notifikasi acara Amazon S3 dengan menggunakan AWS SDK](#)
- [Simpan EXIF dan informasi gambar lainnya menggunakan AWS SDK](#)
- [Kirim pemberitahuan acara S3 ke Amazon EventBridge menggunakan AWS SDK](#)
- [Lacak unggahan atau unduh objek Amazon S3 menggunakan AWS SDK](#)
- [Transformasi data untuk aplikasi Anda dengan S3 Object Lambda](#)
- [Contoh pendekatan untuk pengujian unit dan integrasi dengan AWS SDK](#)
- [Mengunggah direktori lokal secara rekursif ke bucket Amazon Simple Storage Service \(Amazon S3\)](#)
- [Unggah atau unduh file besar ke dan dari Amazon S3 menggunakan AWS SDK](#)
- [Unggah aliran dengan ukuran yang tidak diketahui ke objek Amazon S3 menggunakan AWS SDK](#)
- [Gunakan checksum untuk bekerja dengan objek Amazon S3 menggunakan AWS SDK](#)
- [Bekerja dengan fitur integritas objek Amazon S3 menggunakan AWS SDK](#)
- [Bekerja dengan objek berversi Amazon S3 menggunakan AWS SDK](#)
- [Contoh tanpa server untuk Amazon S3](#)
- [Menginvokasi fungsi Lambda dari pemicu Amazon S3](#)
- [Contoh kode untuk Amazon S3 Control menggunakan AWS SDK](#)

- [Contoh dasar untuk Amazon S3 Control menggunakan AWS SDK](#)
 - [Halo Kontrol Amazon S3](#)
 - [Pelajari dasar-dasar Kontrol Amazon S3 dengan AWS SDK](#)
 - [Tindakan untuk Kontrol Amazon S3 menggunakan AWS SDK](#)
 - [Gunakan CreateJobdengan AWS SDK atau CLI](#)
 - [Gunakan DeleteJobTaggingdengan AWS SDK](#)
 - [Gunakan DescribeJobdengan AWS SDK atau CLI](#)
 - [Gunakan GetJobTaggingdengan AWS SDK](#)
 - [Gunakan PutJobTaggingdengan AWS SDK](#)
 - [Gunakan UpdateJobPrioritydengan AWS SDK atau CLI](#)
 - [Gunakan UpdateJobStatusdengan AWS SDK atau CLI](#)
- [Contoh kode untuk Bucket Direktori S3 menggunakan AWS SDK](#)
- [Contoh dasar untuk Bucket Direktori S3 menggunakan AWS SDK](#)
 - [Halo ember direktori Amazon S3](#)
 - [Pelajari dasar-dasar Bucket Direktori S3 dengan AWS SDK](#)
 - [Tindakan untuk Bucket Direktori S3 menggunakan AWS SDK](#)
 - [Gunakan AbortMultipartUploaddengan AWS SDK](#)
 - [Gunakan CompleteMultipartUploaddengan AWS SDK](#)
 - [Gunakan CopyObjectdengan AWS SDK](#)
 - [Gunakan CreateBucketdengan AWS SDK](#)
 - [Gunakan CreateMultipartUploaddengan AWS SDK](#)
 - [Gunakan CreateSessiondengan AWS SDK](#)
 - [Gunakan DeleteBucketdengan AWS SDK](#)
 - [Gunakan DeleteBucketEncryptiondengan AWS SDK](#)
 - [Gunakan DeleteBucketPolicydengan AWS SDK](#)
 - [Gunakan DeleteObjectdengan AWS SDK](#)
 - [Gunakan DeleteObjectsdengan AWS SDK](#)
 - [Gunakan GetBucketEncryptiondengan AWS SDK](#)
 - [Gunakan GetBucketPolicydengan AWS SDK](#)
 - [Gunakan GetObjectdengan AWS SDK](#)

- [Gunakan GetObjectAttributesdengan AWS SDK](#)
- [Gunakan HeadBucketdengan AWS SDK](#)
- [Gunakan HeadObjectdengan AWS SDK](#)
- [Gunakan ListDirectoryBucketsdengan AWS SDK](#)
- [Gunakan ListMultipartUploadsdengan AWS SDK](#)
- [Gunakan ListObjectsV2 dengan AWS SDK](#)
- [Gunakan ListPartsdengan AWS SDK](#)
- [Gunakan PutBucketEncryptiondengan AWS SDK](#)
- [Gunakan PutBucketPolicydengan AWS SDK](#)
- [Gunakan PutObjectdengan AWS SDK](#)
- [Gunakan UploadPartdengan AWS SDK](#)
- [Gunakan UploadPartCopydengan AWS SDK](#)
- [Skenario untuk Bucket Direktori S3 menggunakan AWS SDK](#)
 - [Buat URL presigned untuk bucket direktori Amazon S3 untuk mendapatkan objek menggunakan AWS SDK](#)

Contoh kode untuk Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan Amazon S3 dengan kit pengembangan AWS perangkat lunak (SDK).

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat[Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Amazon S3

- [Contoh dasar untuk Amazon S3 menggunakan AWS SDK](#)
 - [Halo Amazon S3](#)
 - [Pelajari dasar-dasar Amazon S3 dengan AWS SDK](#)
 - [Tindakan untuk Amazon S3 menggunakan AWS SDK](#)
 - [Gunakan AbortMultipartUploaddengan AWS SDK atau CLI](#)
 - [Gunakan CompleteMultipartUploaddengan AWS SDK atau CLI](#)
 - [Gunakan CopyObjectdengan AWS SDK atau CLI](#)
 - [Gunakan CreateBucketdengan AWS SDK atau CLI](#)
 - [Gunakan CreateMultiRegionAccessPointdengan AWS SDK](#)
 - [Gunakan CreateMultipartUploaddengan AWS SDK atau CLI](#)
 - [Gunakan CreatePresignedPostdengan AWS SDK](#)
 - [Gunakan DeleteBucketdengan AWS SDK atau CLI](#)
 - [Gunakan DeleteBucketAnalyticsConfigurationdengan CLI](#)
 - [Gunakan DeleteBucketCorsdengan AWS SDK atau CLI](#)
 - [Gunakan DeleteBucketEncryptiondengan CLI](#)
 - [Gunakan DeleteBucketInventoryConfigurationdengan CLI](#)
 - [Gunakan DeleteBucketLifecycledengan AWS SDK atau CLI](#)
 - [Gunakan DeleteBucketMetricsConfigurationdengan CLI](#)
 - [Gunakan DeleteBucketPolicydengan AWS SDK atau CLI](#)
 - [Gunakan DeleteBucketReplicationdengan CLI](#)
 - [Gunakan DeleteBucketTaggingdengan CLI](#)
 - [Gunakan DeleteBucketWebsitedengan AWS SDK atau CLI](#)
 - [Gunakan DeleteObjectdengan AWS SDK atau CLI](#)
 - [Gunakan DeleteObjectTaggingdengan CLI](#)
 - [Gunakan DeleteObjectsdengan AWS SDK atau CLI](#)
 - [Gunakan DeletePublicAccessBlockdengan CLI](#)
 - [Gunakan GetBucketAccelerateConfigurationdengan CLI](#)
 - [Gunakan GetBucketAcldengan AWS SDK atau CLI](#)
 - [Gunakan GetBucketAnalyticsConfigurationdengan CLI](#)

- [Gunakan GetBucketCorsdengan AWS SDK atau CLI](#)
- [Gunakan GetBucketEncryptiondengan AWS SDK atau CLI](#)
- [Gunakan GetBucketInventoryConfigurationdengan CLI](#)
- [Gunakan GetBucketLifecycleConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan GetBucketLocationdengan AWS SDK atau CLI](#)
- [Gunakan GetBucketLoggingdengan CLI](#)
- [Gunakan GetBucketMetricsConfigurationdengan CLI](#)
- [Gunakan GetBucketNotificationdengan CLI](#)
- [Gunakan GetBucketPolicydengan AWS SDK atau CLI](#)
- [Gunakan GetBucketPolicyStatusdengan CLI](#)
- [Gunakan GetBucketReplicationdengan AWS SDK atau CLI](#)
- [Gunakan GetBucketRequestPaymentdengan CLI](#)
- [Gunakan GetBucketTaggingdengan CLI](#)
- [Gunakan GetBucketVersioningdengan CLI](#)
- [Gunakan GetBucketWebsitedengan AWS SDK atau CLI](#)
- [Gunakan GetObjectdengan AWS SDK atau CLI](#)
- [Gunakan GetObjectAcldengan AWS SDK atau CLI](#)
- [Gunakan GetObjectAttributesdengan AWS SDK atau CLI](#)
- [Gunakan GetObjectLegalHolddengan AWS SDK atau CLI](#)
- [Gunakan GetObjectLockConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan GetObjectRetentiondengan AWS SDK atau CLI](#)
- [Gunakan GetObjectTaggingdengan CLI](#)
- [Gunakan GetPublicAccessBlockdengan CLI](#)
- [Gunakan HeadBucketdengan AWS SDK atau CLI](#)
- [Gunakan HeadObjectdengan AWS SDK atau CLI](#)
- [Gunakan ListBucketAnalyticsConfigurationsdengan CLI](#)
- [Gunakan ListBucketInventoryConfigurationsdengan CLI](#)
- [Gunakan ListBucketsdengan AWS SDK atau CLI](#)
- [Gunakan ListMultipartUploadsdengan AWS SDK atau CLI](#)
- [Gunakan ListObjectVersionsdengan AWS SDK atau CLI](#)

- [Gunakan ListObjectsdengan CLI](#)
- [Gunakan ListObjectsV2 dengan AWS SDK atau CLI](#)
- [Gunakan PutBucketAccelerateConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketAcldengan AWS SDK atau CLI](#)
- [Gunakan PutBucketCorsdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketEncryptiondengan AWS SDK atau CLI](#)
- [Gunakan PutBucketLifecycleConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketLoggingdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketNotificationdengan CLI](#)
- [Gunakan PutBucketNotificationConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketPolicydengan AWS SDK atau CLI](#)
- [Gunakan PutBucketReplicationdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketRequestPaymentdengan CLI](#)
- [Gunakan PutBucketTaggingdengan CLI](#)
- [Gunakan PutBucketVersioningdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketWebsitedengan AWS SDK atau CLI](#)
- [Gunakan PutObjectdengan AWS SDK atau CLI](#)
- [Gunakan PutObjectAcldengan AWS SDK atau CLI](#)
- [Gunakan PutObjectLegalHolddengan AWS SDK atau CLI](#)
- [Gunakan PutObjectLockConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan PutObjectRetentiondengan AWS SDK atau CLI](#)
- [Gunakan RestoreObjectdengan AWS SDK atau CLI](#)
- [Gunakan SelectObjectContentdengan AWS SDK atau CLI](#)
- [Gunakan UploadPartdengan AWS SDK atau CLI](#)
- [Gunakan UploadPartCopydengan AWS SDK atau CLI](#)
- [Skenario untuk Amazon S3 menggunakan AWS SDK](#)
 - [Periksa apakah ada ember](#)
 - [Periksa apakah ada objek](#)
 - [Mengkonversi teks menjadi ucapan dan kembali ke teks menggunakan AWS SDK](#)
- [Buat URL yang telah ditetapkan sebelumnya untuk Amazon S3 menggunakan AWS SDK](#)

- [Membuat aplikasi manajemen aset foto yang memungkinkan pengguna mengelola foto menggunakan label](#)
- [Halaman web yang mencantumkan objek Amazon S3 menggunakan AWS SDK](#)
- [Membuat aplikasi penjelajah Amazon Textract](#)
- [Hapus semua objek dalam bucket Amazon S3 tertentu menggunakan AWS SDK](#)
- [Hapus unggahan multipart yang tidak lengkap ke Amazon S3 menggunakan AWS SDK](#)
- [Mendeteksi APD dalam gambar dengan Amazon Rekognition menggunakan AWS SDK](#)
- [Mendeteksi entitas dalam teks yang diekstraksi dari gambar menggunakan AWS SDK](#)
- [Mendeteksi wajah dalam gambar menggunakan AWS SDK](#)
- [Mendeteksi objek dalam gambar dengan Amazon Rekognition menggunakan AWS SDK](#)
- [Mendeteksi orang dan objek dalam video dengan Amazon Rekognition menggunakan AWS SDK](#)
- [Unduh 'direktori' S3 dari bucket Amazon Simple Storage Service \(Amazon S3\)](#)
- [Mengunduh semua objek di bucket Amazon Simple Storage Service \(Amazon S3\) ke direktori lokal](#)
- [Unduh aliran dengan ukuran yang tidak diketahui dari objek Amazon S3 menggunakan AWS SDK](#)
- [Dapatkan objek Amazon S3 dari Multi-Region Access Point dengan menggunakan AWS SDK](#)
- [Dapatkan objek dari bucket Amazon S3 menggunakan AWS SDK, menentukan header If-Modified-Since](#)
- [Memulai enkripsi untuk objek Amazon S3 menggunakan AWS SDK](#)
- [Memulai tag untuk objek Amazon S3 menggunakan AWS SDK](#)
- [Memulai dengan cluster pemrosesan data besar](#)
- [Memulai dengan manajemen konfigurasi](#)
- [Memulai dengan ekstraksi teks dokumen](#)
- [Memulai dengan toko fitur pembelajaran mesin](#)
- [Memulai dengan penyimpanan objek](#)
- [Memulai dengan analisis kueri](#)
- [Bekerja dengan fitur kunci objek Amazon S3 menggunakan AWS SDK](#)
- [Buat permintaan bersyarat Amazon S3 menggunakan AWS SDK](#)
- [Mengelola daftar kontrol akses \(ACL\) untuk bucket Amazon S3 menggunakan AWS SDK](#)
- [Mengelola pesan Amazon SQS besar menggunakan Amazon S3 dengan AWS SDK](#)

- [Mengelola objek Amazon S3 berversi dalam batch dengan fungsi Lambda menggunakan AWS SDK](#)
- [Mengurai URI Amazon S3 menggunakan AWS SDK](#)
- [Melakukan salinan multipart dari objek Amazon S3 menggunakan AWS SDK](#)
- [Menerima dan memproses notifikasi acara Amazon S3 dengan menggunakan AWS SDK](#)
- [Simpan EXIF dan informasi gambar lainnya menggunakan AWS SDK](#)
- [Kirim pemberitahuan acara S3 ke Amazon EventBridge menggunakan AWS SDK](#)
- [Lacak unggahan atau unduh objek Amazon S3 menggunakan AWS SDK](#)
- [Transformasi data untuk aplikasi Anda dengan S3 Object Lambda](#)
- [Contoh pendekatan untuk pengujian unit dan integrasi dengan AWS SDK](#)
- [Mengunggah direktori lokal secara rekursif ke bucket Amazon Simple Storage Service \(Amazon S3\)](#)
- [Unggah atau unduh file besar ke dan dari Amazon S3 menggunakan AWS SDK](#)
- [Unggah aliran dengan ukuran yang tidak diketahui ke objek Amazon S3 menggunakan AWS SDK](#)
- [Gunakan checksum untuk bekerja dengan objek Amazon S3 menggunakan AWS SDK](#)
- [Bekerja dengan fitur integritas objek Amazon S3 menggunakan AWS SDK](#)
- [Bekerja dengan objek berversi Amazon S3 menggunakan AWS SDK](#)
- [Contoh tanpa server untuk Amazon S3](#)
- [Menginvokasi fungsi Lambda dari pemicu Amazon S3](#)

Contoh dasar untuk Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan dasar-dasar Amazon Simple Storage Service dengan AWS SDK.

Contoh

- [Halo Amazon S3](#)
- [Pelajari dasar-dasar Amazon S3 dengan AWS SDK](#)
- [Tindakan untuk Amazon S3 menggunakan AWS SDK](#)
- [Gunakan AbortMultipartUpload dengan AWS SDK atau CLI](#)
- [Gunakan CompleteMultipartUpload dengan AWS SDK atau CLI](#)

- [Gunakan CopyObjectdengan AWS SDK atau CLI](#)
- [Gunakan CreateBucketdengan AWS SDK atau CLI](#)
- [Gunakan CreateMultiRegionAccessPointdengan AWS SDK](#)
- [Gunakan CreateMultipartUploaddengan AWS SDK atau CLI](#)
- [Gunakan CreatePresignedPostdengan AWS SDK](#)
- [Gunakan DeleteBucketdengan AWS SDK atau CLI](#)
- [Gunakan DeleteBucketAnalyticsConfigurationdengan CLI](#)
- [Gunakan DeleteBucketCorsdengan AWS SDK atau CLI](#)
- [Gunakan DeleteBucketEncryptiondengan CLI](#)
- [Gunakan DeleteBucketInventoryConfigurationdengan CLI](#)
- [Gunakan DeleteBucketLifecycledengan AWS SDK atau CLI](#)
- [Gunakan DeleteBucketMetricsConfigurationdengan CLI](#)
- [Gunakan DeleteBucketPolicydengan AWS SDK atau CLI](#)
- [Gunakan DeleteBucketReplicationdengan CLI](#)
- [Gunakan DeleteBucketTaggingdengan CLI](#)
- [Gunakan DeleteBucketWebsitedengan AWS SDK atau CLI](#)
- [Gunakan DeleteObjectdengan AWS SDK atau CLI](#)
- [Gunakan DeleteObjectTaggingdengan CLI](#)
- [Gunakan DeleteObjectsdengan AWS SDK atau CLI](#)
- [Gunakan DeletePublicAccessBlockdengan CLI](#)
- [Gunakan GetBucketAccelerateConfigurationdengan CLI](#)
- [Gunakan GetBucketAcldengan AWS SDK atau CLI](#)
- [Gunakan GetBucketAnalyticsConfigurationdengan CLI](#)
- [Gunakan GetBucketCorsdengan AWS SDK atau CLI](#)
- [Gunakan GetBucketEncryptiondengan AWS SDK atau CLI](#)
- [Gunakan GetBucketInventoryConfigurationdengan CLI](#)
- [Gunakan GetBucketLifecycleConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan GetBucketLocationdengan AWS SDK atau CLI](#)
- [Gunakan GetBucketLoggingdengan CLI](#)
- [Gunakan GetBucketMetricsConfigurationdengan CLI](#)

- [Gunakan GetBucketNotificationdengan CLI](#)
- [Gunakan GetBucketPolicydengan AWS SDK atau CLI](#)
- [Gunakan GetBucketPolicyStatusdengan CLI](#)
- [Gunakan GetBucketReplicationdengan AWS SDK atau CLI](#)
- [Gunakan GetBucketRequestPaymentdengan CLI](#)
- [Gunakan GetBucketTaggingdengan CLI](#)
- [Gunakan GetBucketVersioningdengan CLI](#)
- [Gunakan GetBucketWebsitedengan AWS SDK atau CLI](#)
- [Gunakan GetObjectdengan AWS SDK atau CLI](#)
- [Gunakan GetObjectAcldengan AWS SDK atau CLI](#)
- [Gunakan GetObjectAttributesdengan AWS SDK atau CLI](#)
- [Gunakan GetObjectLegalHolddengan AWS SDK atau CLI](#)
- [Gunakan GetObjectLockConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan GetObjectRetentiondengan AWS SDK atau CLI](#)
- [Gunakan GetObjectTaggingdengan CLI](#)
- [Gunakan GetPublicAccessBlockdengan CLI](#)
- [Gunakan HeadBucketdengan AWS SDK atau CLI](#)
- [Gunakan HeadObjectdengan AWS SDK atau CLI](#)
- [Gunakan ListBucketAnalyticsConfigurationsdengan CLI](#)
- [Gunakan ListBucketInventoryConfigurationsdengan CLI](#)
- [Gunakan ListBucketsdengan AWS SDK atau CLI](#)
- [Gunakan ListMultipartUploadsdengan AWS SDK atau CLI](#)
- [Gunakan ListObjectVersionsdengan AWS SDK atau CLI](#)
- [Gunakan ListObjectsdengan CLI](#)
- [Gunakan ListObjectsV2 dengan AWS SDK atau CLI](#)
- [Gunakan PutBucketAccelerateConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketAcldengan AWS SDK atau CLI](#)
- [Gunakan PutBucketCorsdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketEncryptiondengan AWS SDK atau CLI](#)
- [Gunakan PutBucketLifecycleConfigurationdengan AWS SDK atau CLI](#)

- [Gunakan PutBucketLoggingdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketNotificationdengan CLI](#)
- [Gunakan PutBucketNotificationConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketPolicydengan AWS SDK atau CLI](#)
- [Gunakan PutBucketReplicationdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketRequestPaymentdengan CLI](#)
- [Gunakan PutBucketTaggingdengan CLI](#)
- [Gunakan PutBucketVersioningdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketWebsitedengan AWS SDK atau CLI](#)
- [Gunakan PutObjectdengan AWS SDK atau CLI](#)
- [Gunakan PutObjectAcldengan AWS SDK atau CLI](#)
- [Gunakan PutObjectLegalHolddengan AWS SDK atau CLI](#)
- [Gunakan PutObjectLockConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan PutObjectRetentiondengan AWS SDK atau CLI](#)
- [Gunakan RestoreObjectdengan AWS SDK atau CLI](#)
- [Gunakan SelectObjectContentdengan AWS SDK atau CLI](#)
- [Gunakan UploadPartdengan AWS SDK atau CLI](#)
- [Gunakan UploadPartCopydengan AWS SDK atau CLI](#)

Halo Amazon S3

Contoh kode berikut ini menunjukkan cara memulai menggunakan Amazon S3.

.NET

SDK untuk .NET (v4)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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 buckets found.");
            }
        }
        catch { }
    }
}
```

```
        {
            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}");
    }
}
}
```

- Untuk detail API, lihat [ListBuckets](#) di Referensi AWS SDK untuk .NET API.

C++

SDK untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Kode untuk 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})

```

Kode untuk file sumber 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;
}

```

- Untuk detail API, lihat [ListBuckets](#) di Referensi AWS SDK untuk C++ API.

Go

SDK untuk Go V2

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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\t%v\n", *bucket.Name)
    }
}
```

- Untuk detail API, lihat [ListBuckets](#) di Referensi AWS SDK untuk Go API.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
    }
}
}

```

- Untuk detail API, lihat [ListBuckets](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
  }
}
};

```

- Untuk detail API, lihat [ListBuckets](#) di Referensi AWS SDK untuk JavaScript API.

PHP

SDK untuk PHP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

```

use Aws\S3\S3Client;

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

```

- Untuk detail API, lihat [ListBuckets](#) di Referensi AWS SDK untuk PHP API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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()

```

- Untuk detail API, lihat [ListBuckets](#) di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

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

```

```

    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
  end
rescue Aws::Errors::ServiceError => e
  @logger.error("Encountered an error while listing buckets: #{e.message}")
end

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

```

- Untuk detail API, lihat [ListBuckets](#) di Referensi AWS SDK untuk Ruby API.

Rust

SDK for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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 utilitas.

```

/// 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)
    }
}

```

- Untuk detail API, lihat [ListBuckets](#) referensi AWS SDK for Rust API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Pelajari dasar-dasar Amazon S3 dengan AWS SDK

Contoh-contoh kode berikut menunjukkan cara:

- Membuat bucket dan mengunggah file ke dalamnya.
- Mengunduh objek dari bucket.
- Menyalin objek ke subfolder di bucket.
- Membuat daftar objek dalam bucket.
- Menghapus objek bucket dan bucket tersebut.

.NET

SDK untuk .NET (v4)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Jalankan skenario interaktif yang menunjukkan fitur 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.Write("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.Write("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);
                    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;
    }
}
```

Kelas pembungkus untuk metode Amazon S3 SDK.

```
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($"{filePath}\
        \{objectName}", true, CancellationTokens.None);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"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;
    }
}
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk .NET .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Bash

AWS CLI dengan skrip Bash

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
```

Fungsi Amazon S3 yang digunakan dalam skenario ini.

```
#####
# 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
            ;;
        esac
    done
```

```
        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
}

```

```
#####
# 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
}

```

- Untuk detail API, lihat topik berikut di Referensi Perintah AWS CLI .

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

C++

SDK untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di [Repositori Contoh Kode 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();
}

```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk C++ .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Go

SDK untuk Go V2

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Tentukan struktur yang membungkus tindakan bucket dan objek yang digunakan oleh skenario.

```
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
}

```

Jalankan skenario interaktif yang menunjukkan cara bekerja dengan bucket dan objek 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))
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk Go .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Contoh skenario.

```
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.");
        }
    }
}

```

Kelas pembungkus yang berisi operasi.

```

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;
}
}

```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .
 - [CopyObject](#)
 - [CreateBucket](#)

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

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Pertama, impor semua modul yang diperlukan.

```
// 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";
```


Impor sebelumnya merujuk beberapa utilitas pembantu. Utilitas ini bersifat lokal ke GitHub repositori yang ditautkan di awal bagian ini. Untuk referensi, lihat implementasi utilitas tersebut berikut ini.

```
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`;
};

```

Objek disimpan dalam 'ember'. Mari kita tentukan fungsi untuk membuat bucket baru.

```

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;
};

```

Bucket berisi 'objek'. Fungsi ini mengunggah isi direktori ke bucket Anda sebagai objek.

```

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.`);
  }
};
```

Setelah mengunggah objek, pastikan bahwa objek tersebut diunggah dengan benar. Anda dapat menggunakannya ListObjects untuk itu. Anda akan menggunakan properti 'Kunci', tetapi ada juga properti lain yang berguna dalam respons.

```
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`);
};
```

Terkadang Anda perlu menyalin objek dari satu bucket ke bucket lainnya. Gunakan CopyObject perintah untuk itu.

```
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();
};
```

Tidak ada metode SDK yang bisa digunakan untuk mendapatkan beberapa objek dari sebuah bucket. Tetapi, Anda akan membuat daftar objek yang akan diunduh dan mengulanginya.

```
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");
};
```

Saatnya membersihkan sumber daya Anda. Bucket harus kosong sebelum dapat dihapus. Kedua fungsi ini mengosongkan dan menghapus 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`);
};
```

Fungsi 'utama' menyatukan semuanya. Jika Anda menjalankan file ini secara langsung, fungsi utama akan dipanggil.

```
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);
    }
};
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk JavaScript .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)

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

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di [Repositori Contoh Kode 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.
getObjectFromMap(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).asByteArray()
    }

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!")
    }
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK For Kotlin.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

PHP

SDK untuk PHP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di [Repositori Contoh Kode 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";
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk PHP .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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"))
```

- Untuk detail API, lihat topik berikut ini adalah Referensi API SDK untuk Python (Boto3)AWS

.

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Ruby

SDK untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di [Repositori Contoh Kode 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
end
```

```
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}.")
end
```

```
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
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
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__
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk Ruby .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Rust

SDK for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Kode untuk peti biner yang menjalankan skenario.

```
#![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(())
    }
}

```

Tindakan umum yang digunakan oleh skenario.

```

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))
            }
        }
    }
}
}

```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Rust.

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

SAP ABAP

SDK for SAP ABAP

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.

```

- Untuk mengetahui hal detail mengenai API, silakan lihat topik-topik berikut di referensi API AWS SDK untuk ABAP SAP.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)

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

Swift

SDK para Swift

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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)
        }
    }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk Swift.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)

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

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Tindakan untuk Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara melakukan tindakan Amazon S3 individual dengan AWS SDK. Setiap contoh menyertakan tautan ke GitHub, di mana Anda dapat menemukan instruksi untuk mengatur dan menjalankan kode.

Kutipan ini memanggil Amazon S3 API dan merupakan kutipan kode dari program yang lebih besar yang harus dijalankan dalam konteks. Anda dapat melihat tindakan dalam konteks di [Skenario untuk Amazon S3 menggunakan AWS SDK](#).

Contoh berikut hanya mencakup tindakan yang paling umum digunakan. Untuk daftar lengkapnya, lihat [Referensi API Amazon Simple Storage Service](#).

Contoh

- [Gunakan AbortMultipartUploaddengan AWS SDK atau CLI](#)
- [Gunakan CompleteMultipartUploaddengan AWS SDK atau CLI](#)
- [Gunakan CopyObjectdengan AWS SDK atau CLI](#)
- [Gunakan CreateBucketdengan AWS SDK atau CLI](#)
- [Gunakan CreateMultiRegionAccessPointdengan AWS SDK](#)
- [Gunakan CreateMultipartUploaddengan AWS SDK atau CLI](#)
- [Gunakan CreatePresignedPostdengan AWS SDK](#)
- [Gunakan DeleteBucketdengan AWS SDK atau CLI](#)
- [Gunakan DeleteBucketAnalyticsConfigurationdengan CLI](#)
- [Gunakan DeleteBucketCorsdengan AWS SDK atau CLI](#)
- [Gunakan DeleteBucketEncryptiondengan CLI](#)
- [Gunakan DeleteBucketInventoryConfigurationdengan CLI](#)
- [Gunakan DeleteBucketLifecycledengan AWS SDK atau CLI](#)
- [Gunakan DeleteBucketMetricsConfigurationdengan CLI](#)

- [Gunakan DeleteBucketPolicydengan AWS SDK atau CLI](#)
- [Gunakan DeleteBucketReplicationdengan CLI](#)
- [Gunakan DeleteBucketTaggingdengan CLI](#)
- [Gunakan DeleteBucketWebsitedengan AWS SDK atau CLI](#)
- [Gunakan DeleteObjectdengan AWS SDK atau CLI](#)
- [Gunakan DeleteObjectTaggingdengan CLI](#)
- [Gunakan DeleteObjectsdengan AWS SDK atau CLI](#)
- [Gunakan DeletePublicAccessBlockdengan CLI](#)
- [Gunakan GetBucketAccelerateConfigurationdengan CLI](#)
- [Gunakan GetBucketAcldengan AWS SDK atau CLI](#)
- [Gunakan GetBucketAnalyticsConfigurationdengan CLI](#)
- [Gunakan GetBucketCorsdengan AWS SDK atau CLI](#)
- [Gunakan GetBucketEncryptiondengan AWS SDK atau CLI](#)
- [Gunakan GetBucketInventoryConfigurationdengan CLI](#)
- [Gunakan GetBucketLifecycleConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan GetBucketLocationdengan AWS SDK atau CLI](#)
- [Gunakan GetBucketLoggingdengan CLI](#)
- [Gunakan GetBucketMetricsConfigurationdengan CLI](#)
- [Gunakan GetBucketNotificationdengan CLI](#)
- [Gunakan GetBucketPolicydengan AWS SDK atau CLI](#)
- [Gunakan GetBucketPolicyStatusdengan CLI](#)
- [Gunakan GetBucketReplicationdengan AWS SDK atau CLI](#)
- [Gunakan GetBucketRequestPaymentdengan CLI](#)
- [Gunakan GetBucketTaggingdengan CLI](#)
- [Gunakan GetBucketVersioningdengan CLI](#)
- [Gunakan GetBucketWebsitedengan AWS SDK atau CLI](#)
- [Gunakan GetObjectdengan AWS SDK atau CLI](#)
- [Gunakan GetObjectAcldengan AWS SDK atau CLI](#)
- [Gunakan GetObjectAttributesdengan AWS SDK atau CLI](#)
- [Gunakan GetObjectLegalHolddengan AWS SDK atau CLI](#)

- [Gunakan GetObjectLockConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan GetObjectRetentiondengan AWS SDK atau CLI](#)
- [Gunakan GetObjectTaggingdengan CLI](#)
- [Gunakan GetPublicAccessBlockdengan CLI](#)
- [Gunakan HeadBucketdengan AWS SDK atau CLI](#)
- [Gunakan HeadObjectdengan AWS SDK atau CLI](#)
- [Gunakan ListBucketAnalyticsConfigurationsdengan CLI](#)
- [Gunakan ListBucketInventoryConfigurationsdengan CLI](#)
- [Gunakan ListBucketsdengan AWS SDK atau CLI](#)
- [Gunakan ListMultipartUploadsdengan AWS SDK atau CLI](#)
- [Gunakan ListObjectVersionsdengan AWS SDK atau CLI](#)
- [Gunakan ListObjectsdengan CLI](#)
- [Gunakan ListObjectsV2 dengan AWS SDK atau CLI](#)
- [Gunakan PutBucketAccelerateConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketAcldengan AWS SDK atau CLI](#)
- [Gunakan PutBucketCorsdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketEncryptiondengan AWS SDK atau CLI](#)
- [Gunakan PutBucketLifecycleConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketLoggingdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketNotificationdengan CLI](#)
- [Gunakan PutBucketNotificationConfigurationdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketPolicydengan AWS SDK atau CLI](#)
- [Gunakan PutBucketReplicationdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketRequestPaymentdengan CLI](#)
- [Gunakan PutBucketTaggingdengan CLI](#)
- [Gunakan PutBucketVersioningdengan AWS SDK atau CLI](#)
- [Gunakan PutBucketWebsitedengan AWS SDK atau CLI](#)
- [Gunakan PutObjectdengan AWS SDK atau CLI](#)
- [Gunakan PutObjectAcldengan AWS SDK atau CLI](#)
- [Gunakan PutObjectLegalHolddengan AWS SDK atau CLI](#)

- [Gunakan PutObjectLockConfiguration dengan AWS SDK atau CLI](#)
- [Gunakan PutObjectRetention dengan AWS SDK atau CLI](#)
- [Gunakan RestoreObject dengan AWS SDK atau CLI](#)
- [Gunakan SelectObjectContent dengan AWS SDK atau CLI](#)
- [Gunakan UploadPart dengan AWS SDK atau CLI](#)
- [Gunakan UploadPartCopy dengan AWS SDK atau CLI](#)

Gunakan **AbortMultipartUpload** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan AbortMultipartUpload.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Hapus unggahan multipart yang tidak lengkap](#)
- [Bekerja dengan integritas objek Amazon S3](#)

C++

SDK untuk C++

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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();
}

```

- Untuk detail API, lihat [AbortMultipartUpload](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Untuk membatalkan unggahan multipart yang ditentukan

abort-multipart-upload Perintah berikut membatalkan unggahan multibagian untuk kunci multipart/01 di bucket. amzn-s3-demo-bucket

```

aws s3api abort-multipart-upload \
  --bucket amzn-s3-demo-bucket \
  --key multipart/01 \
  --upload-
id dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZLjF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P

```

ID unggahan yang diperlukan oleh perintah ini adalah output oleh create-multipart-upload dan juga dapat diambil dengan list-multipart-uploads.

- Untuk detail API, lihat [AbortMultipartUpload](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
    }
}

```

- Untuk detail API, lihat [AbortMultipartUpload](#) di Referensi AWS SDK for Java 2.x API.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini membatalkan unggahan multipart yang dibuat lebih awal dari 5 hari yang lalu.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

Contoh 2: Perintah ini membatalkan unggahan multipart yang dibuat lebih awal dari 2 Januari 2014.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"Thursday, January 02, 2014"
```

Contoh 3: Perintah ini membatalkan unggahan multipart yang dibuat lebih awal dari 2 Januari 2014, 10:45:37.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"2014/01/02 10:45:37"
```

- Untuk detail API, lihat [AbortMultipartUpload](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini membatalkan unggahan multipart yang dibuat lebih awal dari 5 hari yang lalu.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

Contoh 2: Perintah ini membatalkan unggahan multipart yang dibuat lebih awal dari 2 Januari 2014.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"Thursday, January 02, 2014"
```

Contoh 3: Perintah ini membatalkan unggahan multipart yang dibuat lebih awal dari 2 Januari 2014, 10:45:37.


```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"2014/01/02 10:45:37"
```

- Untuk detail API, lihat [AbortMultipartUpload](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **CompleteMultipartUpload** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `CompleteMultipartUpload`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Melakukan penyalinan multibagian](#)
- [Gunakan checksum](#)
- [Bekerja dengan integritas objek Amazon S3](#)

C++

SDK untuk C++

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
}

```

- Untuk detail API, lihat [CompleteMultipartUpload](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Perintah berikut menyelesaikan unggahan multipart untuk kunci multipart/01 di bucket: amzn-s3-demo-bucket

```
aws s3api complete-multipart-upload --multipart-upload file://
mpustruct --bucket amzn-s3-demo-bucket --key 'multipart/01' --upload-
id dfRtDYU0WMCCcH43C3WfbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6YjlX.cph0gtNBtJ8P
```

ID unggahan yang diperlukan oleh perintah ini adalah output oleh `create-multipart-upload` dan juga dapat diambil dengan `list-multipart-uploads`.

Opsi unggahan multibagian dalam perintah di atas mengambil struktur JSON yang menjelaskan bagian-bagian dari unggahan multibagian yang harus dipasang kembali ke dalam file lengkap. Dalam contoh ini, `file://` awalan digunakan untuk memuat struktur JSON dari file di folder lokal bernama `mpustruct`

`mpustruct`:

```
{
  "Parts": [
    {
      "ETag": "e868e0f4719e394144ef36531ee6824c",
      "PartNumber": 1
    },
    {
      "ETag": "6bb2b12753d66fe86da4998aa33fffb0",
      "PartNumber": 2
    },
    {
      "ETag": "d0a0112e841abec9c9ec83406f0159c8",
      "PartNumber": 3
    }
  ]
}
```

Nilai ETag untuk setiap bagian yang diunggah adalah output setiap kali Anda mengunggah bagian menggunakan `upload-part` perintah dan juga dapat diambil dengan memanggil `list-parts` atau dihitung dengan mengambil checksum MD5 dari setiap bagian.

Output:

```
{
  "ETag": "\"3944a9f7a4faab7f78788ff6210f63f0-3\"",
  "Bucket": "amzn-s3-demo-bucket",
```

```
"Location": "https://amzn-s3-demo-bucket.s3.amazonaws.com/multipart%2F01",  
"Key": "multipart/01"  
}
```

- Untuk detail API, lihat [CompleteMultipartUpload](#) di Referensi AWS CLI Perintah.

Rust

SDK for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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(),
    );
}
```

- Untuk detail API, lihat [CompleteMultipartUpload](#) referensi AWS SDK for Rust API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **CopyObject** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `CopyObject`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)
- [Memulai enkripsi](#)
- [Memulai dengan penyimpanan objek](#)
- [Buat permintaan bersyarat](#)

.NET

SDK untuk .NET (v4)

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
        }
    }
}

```

- Untuk detail API, lihat [CopyObject](#) di Referensi AWS SDK untuk .NET API.

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Salin objek menggunakan permintaan bersyarat.

```

    /// <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;
}
}
}

```

- Untuk detail API, lihat [CopyObject](#) di Referensi AWS SDK untuk .NET API.

Bash

AWS CLI dengan skrip Bash

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}

```

- Untuk detail API, lihat [CopyObject](#) di Referensi AWS CLI Perintah.

C++

SDK untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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();
    }
}
```

- Untuk detail API, lihat [CopyObject](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Perintah berikut menyalin objek dari bucket-1 kebucket-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"  
}
```

- Untuk detail API, lihat [CopyObject](#) di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat [CopyObject](#) di Referensi AWS SDK untuk Go API.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Salin objek menggunakan [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);
    }

```

Gunakan [S3TransferManager](#) untuk [menyalin objek](#) dari satu ember ke ember lainnya. Lihat [file lengkap](#) dan [lakukan pengujian](#).

```

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();
}
```

- Untuk detail API, lihat [CopyObject](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Salin objek.

```
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;
  }
}
};
```

Salin objek dengan syarat ETag tidak cocok dengan yang disediakan.

```
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"));
  }
}
```

Salin objek dengan syarat ETag tidak cocok dengan yang disediakan.

```
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"));
}
}
```

Salin objek menggunakan dengan syarat telah dibuat atau dimodifikasi dalam jangka waktu tertentu.

```
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"));
  }
}

```

Salin objek menggunakan kondisi belum dibuat atau dimodifikasi dalam jangka waktu tertentu.

```

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"));
  }
}
```

- Untuk detail API, lihat [CopyObject](#) di Referensi AWS SDK untuk JavaScript API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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)
    }
}
```

- Untuk detail API, lihat [CopyObject](#) di AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Salinan sederhana dari suatu objek.

```
$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.");
}
```

- Untuk detail API, lihat [CopyObject](#) di Referensi AWS SDK untuk PHP API.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini menyalin objek "sample.txt" dari bucket "test-files" ke bucket yang sama tetapi dengan kunci baru "sample-copy.txt".

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey
sample-copy.txt
```

Contoh 2: Perintah ini menyalin objek "sample.txt" dari bucket "test-files" ke bucket "backup files" dengan kunci "sample-copy.txt".

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -
DestinationKey sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

Contoh 3: Perintah ini mengunduh objek "sample.txt" dari bucket "test-files" ke file lokal dengan nama "local-sample.txt".

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-
sample.txt
```

Contoh 4: Mengunduh objek tunggal ke file yang ditentukan. File yang diunduh akan ditemukan di c:\downloads\data\archive.zip

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder c:\downloads
```

Contoh 5: Download semua objek yang cocok dengan key prefix yang ditentukan ke folder lokal. Hirarki kunci relatif akan dipertahankan sebagai subfolder di lokasi unduhan keseluruhan.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:\downloads
```

- Untuk detail API, lihat [CopyObject](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini menyalin objek "sample.txt" dari bucket "test-files" ke bucket yang sama tetapi dengan kunci baru "sample-copy.txt".

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey sample-copy.txt
```

Contoh 2: Perintah ini menyalin objek "sample.txt" dari bucket "test-files" ke bucket "backup files" dengan kunci "sample-copy.txt".

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -DestinationKey sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

Contoh 3: Perintah ini mengunduh objek "sample.txt" dari bucket "test-files" ke file lokal dengan nama "local-sample.txt".

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-sample.txt
```

Contoh 4: Mengunduh objek tunggal ke file yang ditentukan. File yang diunduh akan ditemukan di c:\downloads\data/archive.zip

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder c:\downloads
```

Contoh 5: Download semua objek yang cocok dengan key prefix yang ditentukan ke folder lokal. Hirarki kunci relatif akan dipertahankan sebagai subfolder di lokasi unduhan keseluruhan.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:\downloads
```

Contoh 6: Mengunduh satu objek ke file lokal menggunakan unduhan paralel multibagian untuk meningkatkan kinerja pada file besar.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-sample.txt -UseMultipartDownload
```

Contoh 7: Mengunduh objek besar menggunakan mode RANGE dengan ukuran bagian 64MB khusus. Mode RANGE menggunakan permintaan rentang byte HTTP dan berfungsi untuk semua objek terlepas dari bagaimana mereka diunggah.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile local-large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -PartSize 64MB
```

Contoh 8: Mengunduh semua objek yang cocok dengan awalan key ke folder lokal menggunakan unduhan multipart dengan unduhan file bersamaan.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:\local-data -UseMultipartDownload -DownloadFilesConcurrently
```

Contoh 9: Mengunduh satu objek menggunakan unduhan multibagian dengan 20 koneksi bersamaan untuk throughput maksimum pada jaringan bandwidth tinggi.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile local-large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

Contoh 10: Mengunduh semua objek yang cocok dengan awalan key ke folder lokal menggunakan unduhan multibagian, melanjutkan kegagalan file individual, bukan membatalkan.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:\local-data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- Untuk detail API, lihat [CopyObject](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

```

Salin objek menggunakan permintaan bersyarat.

```

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

```

- Untuk detail API, lihat [CopyObject](#) di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Salin objek.

```
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
  #                                     copy actions.
  def initialize(source_object)
    @source_object = source_object
  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
```

```

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__

```

Salin objek dan tambahkan enkripsi di sisi server ke objek tujuan.

```

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

  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__

```

- Untuk detail API, lihat [CopyObject](#) di Referensi AWS SDK untuk Ruby API.

Rust

SDK for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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(())
}
```

- Untuk detail API, lihat [CopyObject](#) referensi AWS SDK for Rust API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [CopyObject](#) di AWS SDK untuk referensi SAP ABAP API.

Swift

SDK para Swift

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
import AWSS3  
  
public func copyFile(from sourceBucket: String, name: String, to destBucket:  
String) async throws {
```

```
let srcUrl = ("\"(sourceBucket)/\n(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)\n\nlet input = CopyObjectInput(\n    bucket: destBucket,\n    copySource: srcUrl,\n    key: name\n)\ndo {\n    _ = try await client.copyObject(input: input)\n}\ncatch {\n    print("ERROR: ", dump(error, name: "Copying an object."))\n    throw error\n}\n}
```

- Untuk detail API, lihat referensi [CopyObject AWS SDK](#) untuk Swift API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **CreateBucket** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `CreateBucket`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)
- [Memulai dengan manajemen konfigurasi](#)
- [Memulai dengan toko fitur pembelajaran mesin](#)
- [Memulai dengan penyimpanan objek](#)
- [Kelola pesan besar menggunakan S3](#)
- [Bekerja dengan objek berversi](#)

.NET

SDK untuk .NET (v4)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
    }
}
```

- Untuk detail API, lihat [CreateBucket](#) di Referensi AWS SDK untuk .NET API.

SDK untuk .NET

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat ember dengan kunci objek diaktifkan.

```
/// <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;
    }
}
```

- Untuk detail API, lihat [CreateBucket](#) di Referensi AWS SDK untuk .NET API.

Bash

AWS CLI dengan skrip Bash

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
                ;;
            \?)
                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
}

```

```
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
}
```

- Untuk detail API, lihat [CreateBucket](#) di Referensi AWS CLI Perintah.

C++

SDK untuk C++

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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();
}
```

- Untuk detail API, lihat [CreateBucket](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Contoh 1: Untuk membuat ember

create-bucket Contoh berikut membuat bucket bernama `amzn-s3-demo-bucket`:

```
aws s3api create-bucket \
  --bucket amzn-s3-demo-bucket \
  --region us-east-1
```

Output:

```
{
  "Location": "/amzn-s3-demo-bucket"
}
```

Lihat informasi yang lebih lengkap di [Membuat bucket](#) dalam Panduan Pengguna Amazon S3.

Contoh 2: Untuk membuat ember dengan pemilik diberlakukan

create-bucket Contoh berikut membuat bucket bernama yang menggunakan setelan `amzn-s3-demo-bucket` yang diterapkan pemilik bucket untuk Kepemilikan Objek S3.

```
aws s3api create-bucket \
  --bucket amzn-s3-demo-bucket \
  --region us-east-1 \
  --object-ownership BucketOwnerEnforced
```

Output:

```
{
  "Location": "/amzn-s3-demo-bucket"
}
```

Untuk informasi selengkapnya, lihat [Mengontrol kepemilikan objek dan menonaktifkan ACL di Panduan](#) Pengguna Amazon S3.

Contoh 3: Untuk membuat bucket di luar wilayah ``us-east-1``

`create-bucket` Contoh berikut membuat bucket bernama `amzn-s3-demo-bucket` di `eu-west-1` wilayah tersebut. Daerah di luar `us-east-1` memerlukan yang sesuai `LocationConstraint` untuk ditentukan untuk membuat ember di wilayah yang diinginkan.

```
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/"  
}
```

Lihat informasi yang lebih lengkap di [Membuat bucket](#) dalam Panduan Pengguna Amazon S3.

- Untuk detail API, lihat [CreateBucket](#) di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat bucket dengan konfigurasi default.

```
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  
}
```

Buat ember dengan penguncian objek dan tunggu sampai ada.

```
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
}
```

- Untuk detail API, lihat [CreateBucket](#) di Referensi AWS SDK untuk Go API.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat 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);
        }
    });
}

```

Buat ember dengan kunci objek diaktifkan.

```
// 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");
}
```

- Untuk detail API, lihat [CreateBucket](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat 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;
    }
  }
};
```

- Untuk informasi selengkapnya, lihat [AWS SDK untuk JavaScript Panduan Developer](#).

- Untuk detail API, lihat [CreateBucket](#) di Referensi AWS SDK untuk JavaScript API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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")  
    }  
}
```

- Untuk detail API, lihat [CreateBucket](#) di AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat 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.");
}

```

- Untuk detail API, lihat [CreateBucket](#) di Referensi AWS SDK untuk PHP API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat bucket dengan pengaturan 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

```

Buat bucket berversi dengan konfigurasi siklus hidup.

```

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
```

- Untuk detail API, lihat [CreateBucket](#) di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

# 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__
```

- Untuk detail API, lihat [CreateBucket](#) di Referensi AWS SDK untuk Ruby API.

Rust

SDK for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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))
        }
    })
}

```

- Untuk detail API, lihat [CreateBucket](#) referensi AWS SDK for Rust API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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_bucketalrdyexists.
        MESSAGE 'Bucket name already exists.' TYPE 'E'.
    CATCH /aws1/cx_s3_bktalrdyownedbyyou.
        MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
ENDTRY.

```

- Untuk detail API, lihat [CreateBucket](#) di AWS SDK untuk referensi SAP ABAP API.

Swift

SDK para Swift

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
        }
    }
```

- Untuk detail API, lihat referensi [CreateBucket AWS SDK](#) untuk Swift API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **CreateMultiRegionAccessPoint** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `CreateMultiRegionAccessPoint`.

Kotlin

SDK untuk Kotlin

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Konfigurasi klien kontrol S3 untuk mengirim permintaan ke Wilayah 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
}
```

Buat titik Multi-Region akses.

```

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
                    },
                    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:::$accountIdParam:accesspoint/$mrapAlias"
}

```


Tunggu sampai titik Multi-Region akses tersedia.

```
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)
    }
}
```

- Untuk informasi selengkapnya, lihat [AWS panduan pengembang SDK untuk Kotlin](#).
- Untuk detail API, lihat [CreateMultiRegionAccessPoint](#) di AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **CreateMultipartUpload** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan **CreateMultipartUpload**.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Melakukan penyalinan multibagian](#)
- [Gunakan checksum](#)
- [Bekerja dengan integritas objek Amazon S3](#)

C++

SDK untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
}

```

- Untuk detail API, lihat [CreateMultipartUpload](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Perintah berikut membuat unggahan multipart di bucket `amzn-s3-demo-bucket` dengan kunci `multipart/01`:

```
aws s3api create-multipart-upload --bucket amzn-s3-demo-bucket --key
'multipart/01'
```

Output:

```

{
  "Bucket": "amzn-s3-demo-bucket",
  "UploadId":
    "dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6YjlX.cph0gtNBtJ8P3
  "Key": "multipart/01"
}

```

File yang sudah selesai akan diberi nama `01` dalam folder yang disebut `multipart` dalam ember `amzn-s3-demo-bucket`. Simpan ID unggah, kunci, dan nama bucket untuk digunakan dengan `upload-part` perintah.

- Untuk detail API, lihat [CreateMultipartUpload](#) di Referensi AWS CLI Perintah.

Rust

SDK for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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?;

```

- Untuk detail API, lihat [CreateMultipartUpload](#) referensi AWS SDK for Rust API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **CreatePresignedPost** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan **CreatePresignedPost**.

.NET

SDK untuk .NET (v4)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat URL POST yang telah ditentukan sebelumnya.

```
/// <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);
}
```

- Untuk detail API, lihat [CreatePresignedPost](#) di Referensi AWS SDK untuk .NET API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteBucket** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DeleteBucket`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)
- [Memulai dengan manajemen konfigurasi](#)
- [Memulai dengan toko fitur pembelajaran mesin](#)
- [Memulai dengan penyimpanan objek](#)

.NET

SDK untuk .NET (v4)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
    }
}
```

- Untuk detail API, lihat [DeleteBucket](#) di Referensi AWS SDK untuk .NET API.

Bash

AWS CLI dengan skrip Bash

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat [DeleteBucket](#) di Referensi AWS CLI Perintah.

C++

SDK untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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();
}
```

- Untuk detail API, lihat [DeleteBucket](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Perintah berikut menghapus bucket bernama `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket --bucket amzn-s3-demo-bucket --region us-east-1
```

- Untuk detail API, lihat [DeleteBucket](#) di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat [DeleteBucket](#) di Referensi AWS SDK untuk Go API.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
}
```

- Untuk detail API, lihat [DeleteBucket](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hapus 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;  
  }  
}  
};
```

- Untuk informasi selengkapnya, lihat [AWS SDK untuk JavaScript Panduan Developer](#).
- Untuk detail API, lihat [DeleteBucket](#) di Referensi AWS SDK untuk JavaScript API.

PHP

SDK untuk PHP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hapus bucket kosong.

```
$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.");  
}
```

- Untuk detail API, lihat [DeleteBucket](#) di Referensi AWS SDK untuk PHP API.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini menghapus semua objek dan versi objek dari bucket 'test-files' dan kemudian menghapus bucket. Perintah akan meminta konfirmasi sebelum melanjutkan. Tambahkan sakelar -Force untuk menekan konfirmasi. Perhatikan bahwa ember yang tidak kosong tidak dapat dihapus.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- Untuk detail API, lihat [DeleteBucket](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini menghapus semua objek dan versi objek dari bucket 'test-files' dan kemudian menghapus bucket. Perintah akan meminta konfirmasi sebelum melanjutkan. Tambahkan sakelar -Force untuk menekan konfirmasi. Perhatikan bahwa ember yang tidak kosong tidak dapat dihapus.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- Untuk detail API, lihat [DeleteBucket](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

```

- Untuk detail API, lihat [DeleteBucket](#) di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

```

- Untuk detail API, lihat [DeleteBucket](#) di Referensi AWS SDK untuk Ruby API.

Rust

SDK for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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))
            }
        }
    }
}

```

```
}
```

- Untuk detail API, lihat [DeleteBucket](#) referensi AWS SDK for Rust API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [DeleteBucket](#) di AWS SDK untuk referensi SAP ABAP API.

Swift

SDK para Swift

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
    }
}

```

- Untuk detail API, lihat referensi [DeleteBucket AWSSDK](#) untuk Swift API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteBucketAnalyticsConfiguration** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan **DeleteBucketAnalyticsConfiguration**.

CLI

AWS CLI

Untuk menghapus konfigurasi analitik untuk bucket

delete-bucket-analytics-configuration Contoh berikut menghapus konfigurasi analitik untuk bucket dan ID yang ditentukan.

```

aws s3api delete-bucket-analytics-configuration \
    --bucket amzn-s3-demo-bucket \
    --id 1

```

Perintah ini tidak menghasilkan output.

- Untuk detail API, lihat [DeleteBucketAnalyticsConfiguration](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah menghapus filter analitik dengan nama 'testfilter' di bucket S3 yang diberikan.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AnalyticsId 'testfilter'
```

- Untuk detail API, lihat [DeleteBucketAnalyticsConfiguration](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah menghapus filter analitik dengan nama 'testfilter' di bucket S3 yang diberikan.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AnalyticsId 'testfilter'
```

- Untuk detail API, lihat [DeleteBucketAnalyticsConfiguration](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteBucketCors** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DeleteBucketCors`.

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
}
```

- Untuk detail API, lihat [DeleteBucketCors](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Perintah berikut menghapus konfigurasi Berbagi Cross-Origin Sumber Daya dari bucket bernama `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-cors --bucket amzn-s3-demo-bucket
```

- Untuk detail API, lihat [DeleteBucketCors](#) di Referensi AWS CLI Perintah.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
```

- Untuk detail API, lihat [DeleteBucketCors](#) di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

end
```

- Untuk detail API, lihat [DeleteBucketCors](#) di Referensi AWS SDK untuk Ruby API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [DeleteBucketCors](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteBucketEncryption** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan **DeleteBucketEncryption**.

CLI

AWS CLI

Untuk menghapus konfigurasi enkripsi sisi server dari bucket

delete-bucket-encryption Contoh berikut menghapus konfigurasi enkripsi sisi server dari bucket yang ditentukan.

```
aws s3api delete-bucket-encryption \
    --bucket amzn-s3-demo-bucket
```

Perintah ini tidak menghasilkan output.

- Untuk detail API, lihat [DeleteBucketEncryption](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Ini menonaktifkan enkripsi yang diaktifkan untuk bucket S3 yang disediakan.

```
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
```

- Untuk detail API, lihat [DeleteBucketEncryption](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Ini menonaktifkan enkripsi yang diaktifkan untuk bucket S3 yang disediakan.

```
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
```

- Untuk detail API, lihat [DeleteBucketEncryption](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteBucketInventoryConfiguration** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan **DeleteBucketInventoryConfiguration**.

CLI

AWS CLI

Untuk menghapus konfigurasi inventaris bucket

`delete-bucket-inventory-configuration` Contoh berikut menghapus konfigurasi inventaris dengan ID 1 untuk bucket yang ditentukan.

```
aws s3api delete-bucket-inventory-configuration \
  --bucket amzn-s3-demo-bucket \
  --id 1
```

Perintah ini tidak menghasilkan output.

- Untuk detail API, lihat [DeleteBucketInventoryConfiguration](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini menghapus inventori bernama InventoryName 'test' yang sesuai dengan bucket S3 yang diberikan.

```
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
```

- Untuk detail API, lihat [DeleteBucketInventoryConfiguration](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini menghapus inventori bernama InventoryName 'test' yang sesuai dengan bucket S3 yang diberikan.

```
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

- Untuk detail API, lihat [DeleteBucketInventoryConfiguration](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteBucketLifecycle** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DeleteBucketLifecycle`.

.NET

SDK untuk .NET

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
}
```

- Untuk detail API, lihat [DeleteBucketLifecycle](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Perintah berikut menghapus konfigurasi siklus hidup dari bucket bernama: `amzn-s3-demo-bucket`

```
aws s3api delete-bucket-lifecycle --bucket amzn-s3-demo-bucket
```

- Untuk detail API, lihat [DeleteBucketLifecycle](#) di Referensi AWS CLI Perintah.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

```

- Untuk detail API, lihat [DeleteBucketLifecycle](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.

```

- Untuk detail API, lihat [DeleteBucketLifecycle](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteBucketMetricsConfiguration** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan **DeleteBucketMetricsConfiguration**.

CLI

AWS CLI

Untuk menghapus konfigurasi metrik untuk bucket

delete-bucket-metrics-configuration Contoh berikut menghapus konfigurasi metrik untuk bucket dan ID yang ditentukan.

```
aws s3api delete-bucket-metrics-configuration \
  --bucket amzn-s3-demo-bucket \
  --id 123
```

Perintah ini tidak menghasilkan output.

- Untuk detail API, lihat [DeleteBucketMetricsConfiguration](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah menghapus filter metrik dengan nama 'testmetrics' di bucket S3 yang diberikan.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
'testmetrics'
```

- Untuk detail API, lihat [DeleteBucketMetricsConfiguration](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah menghapus filter metrik dengan nama 'testmetrics' di bucket S3 yang diberikan.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testmetrics'
```

- Untuk detail API, lihat [DeleteBucketMetricsConfiguration](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteBucketPolicy** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DeleteBucketPolicy`.

C++

SDK untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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();
}

```

- Untuk detail API, lihat [DeleteBucketPolicy](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Perintah berikut menghapus kebijakan bucket dari bucket bernama `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-policy --bucket amzn-s3-demo-bucket
```

- Untuk detail API, lihat [DeleteBucketPolicy](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
    }
}
```

- Untuk detail API, lihat [DeleteBucketPolicy](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hapus kebijakan 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;
    }
  }
};
```

- Untuk informasi selengkapnya, lihat [AWS SDK untuk JavaScript Panduan Developer](#).
- Untuk detail API, lihat [DeleteBucketPolicy](#) di Referensi AWS SDK untuk JavaScript API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {  
    val request =  
        DeleteBucketPolicyRequest {  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.deleteBucketPolicy(request)  
        println("Done!")  
    }  
}
```

- Untuk detail API, lihat [DeleteBucketPolicy](#) di AWS SDK untuk referensi API Kotlin.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah menghapus kebijakan bucket yang terkait dengan bucket S3 yang diberikan.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [DeleteBucketPolicy](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah menghapus kebijakan bucket yang terkait dengan bucket S3 yang diberikan.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [DeleteBucketPolicy](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
```

- Untuk detail API, lihat [DeleteBucketPolicy](#) di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
```

- Untuk detail API, lihat [DeleteBucketPolicy](#) di Referensi AWS SDK untuk Ruby API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [DeleteBucketPolicy](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteBucketReplication** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan `DeleteBucketReplication`.

CLI

AWS CLI

Perintah berikut menghapus konfigurasi replikasi dari bucket bernama: `amzn-s3-demo-bucket`

```
aws s3api delete-bucket-replication --bucket amzn-s3-demo-bucket
```

- Untuk detail API, lihat [DeleteBucketReplication](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Menghapus konfigurasi replikasi yang terkait dengan bucket bernama 'amzn-s3-demo-bucket'. Perhatikan bahwa operasi ini memerlukan izin untuk DeleteReplicationConfiguration tindakan s3:. Anda akan diminta konfirmasi sebelum operasi berlangsung - untuk menekan konfirmasi, gunakan sakelar -Force.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Untuk detail API, lihat [DeleteBucketReplication](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Menghapus konfigurasi replikasi yang terkait dengan bucket bernama 'amzn-s3-demo-bucket'. Perhatikan bahwa operasi ini memerlukan izin untuk DeleteReplicationConfiguration tindakan s3:. Anda akan diminta konfirmasi sebelum operasi berlangsung - untuk menekan konfirmasi, gunakan sakelar -Force.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Untuk detail API, lihat [DeleteBucketReplication](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteBucketTagging** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan DeleteBucketTagging.

CLI

AWS CLI

Perintah berikut menghapus konfigurasi penandaan dari bucket bernama: amzn-s3-demo-bucket

```
aws s3api delete-bucket-tagging --bucket amzn-s3-demo-bucket
```

- Untuk detail API, lihat [DeleteBucketTagging](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini menghapus semua tag yang terkait dengan bucket S3 yang diberikan.

```
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
```

- Untuk detail API, lihat [DeleteBucketTagging](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini menghapus semua tag yang terkait dengan bucket S3 yang diberikan.

```
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
```

- Untuk detail API, lihat [DeleteBucketTagging](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteBucketWebsite** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DeleteBucketWebsite`.

C++

SDK untuk C++

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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();
}
```

- Untuk detail API, lihat [DeleteBucketWebsite](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Perintah berikut menghapus konfigurasi situs web dari bucket bernama `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-website --bucket amzn-s3-demo-bucket
```

- Untuk detail API, lihat [DeleteBucketWebsite](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
    }
}

```

- Untuk detail API, lihat [DeleteBucketWebsite](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hapus konfigurasi situs web dari 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;
    }
}
};
```

- Untuk informasi selengkapnya, lihat [AWS SDK untuk JavaScript Panduan Developer](#).
- Untuk detail API, lihat [DeleteBucketWebsite](#) di Referensi AWS SDK untuk JavaScript API.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini menonaktifkan properti hosting situs web statis dari bucket S3 yang diberikan.

```
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
```

- Untuk detail API, lihat [DeleteBucketWebsite](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini menonaktifkan properti hosting situs web statis dari bucket S3 yang diberikan.

```
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
```

- Untuk detail API, lihat [DeleteBucketWebsite](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteObject** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DeleteObject`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Bekerja dengan integritas objek Amazon S3](#)
- [Bekerja dengan objek berversi](#)

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hapus objek dalam bucket S3 yang tidak berversi.

```
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.");
        }
    }
}

```

Hapus objek dalam bucket S3 berversi.

```

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 = "verstioned-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.USSouth1;
            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 co 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;
}
}

```

- Untuk detail API, lihat [DeleteObject](#) di Referensi AWS SDK untuk .NET API.

Bash

AWS CLI dengan skrip Bash

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}

```

- Untuk detail API, lihat [DeleteObject](#) di Referensi AWS CLI Perintah.

C++

SDK untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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();  
}
```

- Untuk detail API, lihat [DeleteObject](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Perintah berikut menghapus objek bernama `test.txt` dari bucket bernama `amzn-s3-demo-bucket`:

```
aws s3api delete-object --bucket amzn-s3-demo-bucket --key test.txt
```

Jika pembuatan versi bucket diaktifkan, output akan berisi ID versi penanda hapus:

```
{  
  "VersionId": "9_gKg5vG56F.TTEUdwkxGpJ3tND1WlGq",  
  "DeleteMarker": true  
}
```

Untuk informasi selengkapnya tentang menghapus objek, lihat [Menghapus Objek](#) di Panduan Pengembang Amazon S3.

- Untuk detail API, lihat [DeleteObject](#) di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}

```

- Untuk detail API, lihat [DeleteObject](#) di Referensi AWS SDK untuk Go API.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
}

```

- Untuk detail API, lihat [DeleteObject](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hapus objek.

```
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;  
    }  
}  
};
```

- Untuk detail API, lihat [DeleteObject](#) di Referensi AWS SDK untuk JavaScript API.

PHP

SDK untuk PHP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;  
    }  
}
```

- Untuk detail API, lihat [DeleteObject](#) di Referensi AWS SDK untuk PHP API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hapus objek.

```
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

```

Kembalikan objek ke versi sebelumnya dengan menghapus versi yang lebih baru dari objek tersebut.

```

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}."
    )

```

Aktifkan kembali objek yang dihapus dengan menghapus penanda hapus aktif objek.

```

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)

```

Buat penanganan Lambda yang menghapus penanda hapus dari objek S3. Penangan ini dapat digunakan untuk membersihkan penanda hapus asing secara efisien dalam bucket berversi.

```

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,
    }
```

- Untuk detail API, lihat [DeleteObject](#) di AWS SDK for Python (Boto3) Referensi API.

Rust

SDK for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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?;\n\n        // There are no modeled errors to handle when deleting an object.\n\n        Ok(())\n    }\n}
```

- Untuk detail API, lihat [DeleteObject](#) referensi AWS SDK for Rust API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
TRY.\n    lo_s3->deleteobject(\n        iv_bucket = iv_bucket_name\n        iv_key = iv_object_key ).\n    MESSAGE 'Object deleted from S3 bucket.' TYPE 'I'. \n    CATCH /aws1/cx_s3_nosuchbucket.\n        MESSAGE 'Bucket does not exist.' TYPE 'E'. \n    ENDTRY.
```

- Untuk detail API, lihat [DeleteObject](#) di AWS SDK untuk referensi SAP ABAP API.

Swift

SDK para Swift

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
    }
}
```

- Untuk detail API, lihat referensi [DeleteObject AWS](#) SDK untuk Swift API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteObjectTagging** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan `DeleteObjectTagging`.

CLI

AWS CLI

Untuk menghapus set tag dari suatu objek

delete-object-tagging Contoh berikut menghapus tag dengan kunci yang ditentukan dari objek doc1.rtf.

```
aws s3api delete-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc1.rtf
```

Perintah ini tidak menghasilkan output.

- Untuk detail API, lihat [DeleteObjectTagging](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini menghapus semua tag yang terkait dengan objek dengan kunci 'testfile.txt' di Bucket S3 yang diberikan.

```
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
```

- Untuk detail API, lihat [DeleteObjectTagging](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini menghapus semua tag yang terkait dengan objek dengan kunci 'testfile.txt' di Bucket S3 yang diberikan.

```
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
```

- Untuk detail API, lihat [DeleteObjectTagging](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteObjects** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DeleteObjects`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)
- [Hapus semua objek dalam ember](#)
- [Memulai dengan penyimpanan objek](#)

.NET

SDK untuk .NET (v4)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
    }
}
```



```
}
```

- Untuk detail API, lihat [DeleteObjects](#) di Referensi AWS SDK untuk .NET API.

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hapus beberapa objek dalam bucket S3 yang tidak berversi.

```
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();
```

```

        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;
}
}

```

Hapus beberapa objek dalam bucket S3 berversi.

```

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.USSouth1;
        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
        and
        // 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;
}
}

```

- Untuk detail API, lihat [DeleteObjects](#) di Referensi AWS SDK untuk .NET API.

Bash

AWS CLI dengan skrip Bash

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat [DeleteObjects](#) di Referensi AWS CLI Perintah.

C++

SDK untuk C++

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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();
}
```

- Untuk detail API, lihat [DeleteObjects](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Perintah berikut menghapus objek dari bucket bernama `amzn-s3-demo-bucket`:

```
aws s3api delete-objects --bucket amzn-s3-demo-bucket --delete file://delete.json
```

`delete.json` adalah dokumen JSON di direktori saat ini yang menentukan objek yang akan dihapus:

```
{
  "Objects": [
    {
      "Key": "test1.txt"
    }
  ],
  "Quiet": false
}
```

Output:

```
{
  "Deleted": [
    {
      "DeleteMarkerVersionId": "mYAT5Mc6F7aeUL8SS7FAAqUP01koHwzU",
      "Key": "test1.txt",
      "DeleteMarker": true
    }
  ]
}
```

- Untuk detail API, lihat [DeleteObjects](#) di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
```


- Untuk detail API, lihat [DeleteObjects](#) di Referensi AWS SDK untuk Go API.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
        }
    }
}
```

- Untuk detail API, lihat [DeleteObjects](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hapus beberapa objek.

```
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;  
  }  
}  
};
```

- Untuk detail API, lihat [DeleteObjects](#) di Referensi AWS SDK untuk JavaScript API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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")
    }
}
```

- Untuk detail API, lihat [DeleteObjects](#) di AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hapus satu set objek dari daftar kunci.

```
$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.");
    }
```

- Untuk detail API, lihat [DeleteObjects](#) di Referensi AWS SDK untuk PHP API.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini menghapus objek "sample.txt" dari bucket "test-files". Anda diminta konfirmasi sebelum perintah dijalankan; untuk menekan prompt gunakan sakelar -Force.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Contoh 2: Perintah ini menghapus versi objek "sample.txt" yang ditentukan dari bucket "test-files", dengan asumsi bucket telah dikonfigurasi untuk mengaktifkan versi objek.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Contoh 3: Perintah ini menghapus objek "sample1.txt", "sample2.txt" dan "sample3.txt" dari bucket "test-files" sebagai operasi batch tunggal. Respons layanan akan mencantumkan semua kunci yang diproses, terlepas dari status keberhasilan atau kesalahan penghapusan. Untuk mendapatkan hanya kesalahan untuk kunci yang tidak dapat diproses oleh layanan tambahkan ReportErrorsOnly parameter - (parameter ini juga dapat ditentukan dengan alias -Quiet).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
    "sample2.txt", "sample3.txt" )
```

Contoh 4: Contoh ini menggunakan ekspresi sebaris dengan KeyCollection parameter - untuk mendapatkan kunci objek yang akan dihapus. Get-S3Object mengembalikan koleksi Amazon.S3.Model.S3Object contoh, masing-masing memiliki anggota Key dari jenis string mengidentifikasi objek.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Contoh 5: Contoh ini memperoleh semua objek yang memiliki key prefix prefix/subprefix "" di bucket dan menghapusnya. Perhatikan bahwa objek yang masuk diproses satu per satu. Untuk koleksi besar, pertimbangkan untuk meneruskan koleksi ke parameter cmdlet -InputObject (alias -S3ObjectCollection) untuk memungkinkan penghapusan terjadi sebagai batch dengan satu panggilan ke layanan.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |
Remove-S3Object -Force
```

Contoh 6: Contoh ini menyalurkan kumpulan Amazon.S3.Model.S3ObjectVersion instance yang mewakili penanda hapus ke cmdlet untuk dihapus. Perhatikan bahwa objek yang masuk diproses satu per satu. Untuk koleksi besar, pertimbangkan untuk meneruskan koleksi ke parameter cmdlet -InputObject (alias -S3ObjectCollection) untuk memungkinkan penghapusan terjadi sebagai batch dengan satu panggilan ke layanan.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Contoh 7: Script ini menunjukkan bagaimana melakukan penghapusan batch dari satu set objek (dalam hal ini menghapus penanda) dengan membangun array objek yang akan digunakan dengan -KeyAndVersionCollection parameter.

```
$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
```

- Untuk detail API, lihat [DeleteObjects](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini menghapus objek "sample.txt" dari bucket "test-files". Anda diminta konfirmasi sebelum perintah dijalankan; untuk menekan prompt gunakan sakelar -Force.


```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Contoh 2: Perintah ini menghapus versi objek "sample.txt" yang ditentukan dari bucket "test-files", dengan asumsi bucket telah dikonfigurasi untuk mengaktifkan versi objek.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId  
HLbxx6V9omT6AQYVpks8mmFKQcejpqt
```

Contoh 3: Perintah ini menghapus objek "sample1.txt", "sample2.txt" dan "sample3.txt" dari bucket "test-files" sebagai operasi batch tunggal. Respons layanan akan mencantumkan semua kunci yang diproses, terlepas dari status keberhasilan atau kesalahan penghapusan. Untuk mendapatkan hanya kesalahan untuk kunci yang tidak dapat diproses oleh layanan tambahkan ReportErrorsOnly parameter - (parameter ini juga dapat ditentukan dengan alias - Quiet).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",  
"sample2.txt", "sample3.txt" )
```

Contoh 4: Contoh ini menggunakan ekspresi sebaris dengan KeyCollection parameter - untuk mendapatkan kunci objek yang akan dihapus. Get-S3Object mengembalikan koleksi Amazon.S3.Model.S3Object contoh, masing-masing memiliki anggota Key dari jenis string mengidentifikasi objek.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object  
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Contoh 5: Contoh ini memperoleh semua objek yang memiliki key prefix prefix/subprefix "" di bucket dan menghapusnya. Perhatikan bahwa objek yang masuk diproses satu per satu. Untuk koleksi besar, pertimbangkan untuk meneruskan koleksi ke parameter cmdlet - InputObject (alias -S3ObjectCollection) untuk memungkinkan penghapusan terjadi sebagai batch dengan satu panggilan ke layanan.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |  
Remove-S3Object -Force
```

Contoh 6: Contoh ini menyalurkan kumpulan Amazon.S3.Model.S3ObjectVersion instance yang mewakili penanda hapus ke cmdlet untuk dihapus. Perhatikan bahwa objek yang masuk diproses satu per satu. Untuk koleksi besar, pertimbangkan untuk meneruskan koleksi ke

parameter cmdlet - InputObject (alias -S3ObjectCollection) untuk memungkinkan penghapusan terjadi sebagai batch dengan satu panggilan ke layanan.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Contoh 7: Script ini menunjukkan bagaimana melakukan penghapusan batch dari satu set objek (dalam hal ini menghapus penanda) dengan membangun array objek yang akan digunakan dengan - KeyAndVersionCollection parameter.

```
$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
```

- Untuk detail API, lihat [DeleteObjects](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hapus satu set objek dengan menggunakan daftar kunci objek.

```
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

```

Hapus semua objek dalam 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
```

Hapus objek berversi secara permanen dengan menghapus semua versinya.

```
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

```

- Untuk detail API, lihat [DeleteObjects](#) di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

```

- Untuk detail API, lihat [DeleteObjects](#) di Referensi AWS SDK untuk Ruby API.

Rust

SDK for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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(())
}

```

- Untuk detail API, lihat [DeleteObjects](#) referensi AWS SDK for Rust API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.

```

- Untuk detail API, lihat [DeleteObjects](#) di AWS SDK untuk referensi SAP ABAP API.

Swift

SDK para Swift

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
    }
}
```

- Untuk detail API, lihat referensi [DeleteObjects AWS](#) SDK untuk Swift API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeletePublicAccessBlock** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan **DeletePublicAccessBlock**.

CLI

AWS CLI

Untuk menghapus konfigurasi blokir akses publik untuk bucket

`delete-public-access-block` Contoh berikut menghapus konfigurasi blok akses publik pada bucket yang ditentukan.

```
aws s3api delete-public-access-block \
  --bucket amzn-s3-demo-bucket
```

Perintah ini tidak menghasilkan output.

- Untuk detail API, lihat [DeletePublicAccessBlock](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mematikan setelan blokir akses publik untuk bucket yang diberikan.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select
'^BucketName'
```

Output:

```
amzn-s3-demo-bucket
```

- Untuk detail API, lihat [DeletePublicAccessBlock](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mematikan setelan blokir akses publik untuk bucket yang diberikan.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select
'^BucketName'
```

Output:

```
amzn-s3-demo-bucket
```

- Untuk detail API, lihat [DeletePublicAccessBlock](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketAccelerateConfiguration** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan **GetBucketAccelerateConfiguration**.

CLI

AWS CLI

Untuk mengambil konfigurasi percepatan bucket

`get-bucket-accelerate-configuration` Contoh berikut mengambil konfigurasi percepatan untuk bucket yang ditentukan.

```
aws s3api get-bucket-accelerate-configuration \
  --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "Status": "Enabled"
}
```

- Untuk detail API, lihat [GetBucketAccelerateConfiguration](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengembalikan nilai Diaktifkan, jika pengaturan akselerasi transfer diaktifkan untuk bucket yang ditentukan.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Value
-----
Enabled
```

- Untuk detail API, lihat [GetBucketAccelerateConfiguration](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengembalikan nilai Diaktifkan, jika pengaturan akselerasi transfer diaktifkan untuk bucket yang ditentukan.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Value
-----
Enabled
```

- Untuk detail API, lihat [GetBucketAccelerateConfiguration](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketAcl** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `GetBucketAcl`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Periksa apakah ada ember](#)
- [Mengelola daftar kontrol akses \(ACL\)](#)

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
}
```

- Untuk detail API, lihat [GetBucketAcl](#) di Referensi AWS SDK untuk .NET API.

C++

SDK untuk C++

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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";
}

```

- Untuk detail API, lihat [GetBucketAcl](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Perintah berikut mengambil daftar kontrol akses untuk bucket bernama `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"
    }
  ]
}
```

- Untuk detail API, lihat [GetBucketAcl](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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 "";
    }
}
```

- Untuk detail API, lihat [GetBucketAcl](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Dapatkan izin 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;
    }
}
};
```

- Untuk informasi selengkapnya, lihat [AWS SDK untuk JavaScript Panduan Developer](#).
- Untuk detail API, lihat [GetBucketAcl](#) di Referensi AWS SDK untuk JavaScript API.

PowerShell

Alat untuk PowerShell V5

Contoh 1: Perintah mendapatkan rincian pemilik objek objek S3.

```
(Get-S3BucketACL -BucketName 'amzn-s3-demo-bucket' -Select *).Owner
```

Output:

```
DisplayName Id
----- --
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- Untuk detail API, lihat [GetBucketAcl](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
```

- Untuk detail API, lihat [GetBucketAcl](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [GetBucketAcl](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketAnalyticsConfiguration** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan **GetBucketAnalyticsConfiguration**.

CLI

AWS CLI

Untuk mengambil konfigurasi analitik untuk bucket dengan ID tertentu

get-bucket-analytics-configuration Contoh berikut menampilkan konfigurasi analitik untuk bucket dan ID yang ditentukan.

```
aws s3api get-bucket-analytics-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --id 1
```

Output:

```
{  
  "AnalyticsConfiguration": {  
    "StorageClassAnalysis": {},  
    "Id": "1"  
  }  
}
```

- Untuk detail API, lihat [GetBucketAnalyticsConfiguration](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengembalikan detail filter analitik dengan nama 'testfilter' di bucket S3 yang diberikan.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId  
'testfilter'
```

- Untuk detail API, lihat [GetBucketAnalyticsConfiguration](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengembalikan detail filter analitik dengan nama 'testfilter' di bucket S3 yang diberikan.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId  
'testfilter'
```

- Untuk detail API, lihat [GetBucketAnalyticsConfiguration](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketCors** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `GetBucketCors`.

.NET

SDK untuk .NET

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
}
```

- Untuk detail API, lihat [GetBucketCors](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Perintah berikut mengambil konfigurasi Berbagi Cross-Origin Sumber Daya untuk bucket bernama `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": [
        "*"
      ]
    }
  ]
}
```

```
}
```

- Untuk detail API, lihat [GetBucketCors](#) di Referensi AWS CLI Perintah.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Dapatkan kebijakan CORS untuk 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(`S3ServiceException: ${err.message}`);
    } else {
      console.log(`Error: ${err.message}`);
    }
  }
};
```

```

        '\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;
    }
}
};

```

- Untuk informasi selengkapnya, lihat [AWS SDK untuk JavaScript Panduan Developer](#).
- Untuk detail API, lihat [GetBucketCors](#) di Referensi AWS SDK untuk JavaScript API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

```

- Untuk detail API, lihat [GetBucketCors](#) di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
end
```

- Untuk detail API, lihat [GetBucketCors](#) di Referensi AWS SDK untuk Ruby API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

TRY.

```

        oo_result = lo_s3->getbucketcors(
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.

```

- Untuk detail API, lihat [GetBucketCors](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketEncryption** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `GetBucketEncryption`.

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.ServerSideEncryptionKeyManagementServiceId}");
        }
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine(ex.ErrorCode == "InvalidBucketName"
            ? $"Bucket {bucketName} was not found."
            : $"Unable to get bucket encryption for bucket {bucketName},
{ex.Message}");
    }
}

```

- Untuk detail API, lihat [GetBucketEncryption](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Untuk mengambil konfigurasi enkripsi sisi server untuk bucket

get-bucket-encryption Contoh berikut mengambil konfigurasi enkripsi sisi server untuk bucket. `amzn-s3-demo-bucket`

```
aws s3api get-bucket-encryption \
    --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "ServerSideEncryptionConfiguration": {
    "Rules": [
      {
        "ApplyServerSideEncryptionByDefault": {
          "SSEAlgorithm": "AES256"
        }
      }
    ]
  }
}
```

- Untuk detail API, lihat [GetBucketEncryption](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengembalikan semua aturan enkripsi sisi server yang terkait dengan bucket yang diberikan.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [GetBucketEncryption](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengembalikan semua aturan enkripsi sisi server yang terkait dengan bucket yang diberikan.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [GetBucketEncryption](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan `GetBucketInventoryConfiguration` dengan CLI

Contoh kode berikut menunjukkan cara menggunakan `GetBucketInventoryConfiguration`.

CLI

AWS CLI

Untuk mengambil konfigurasi inventaris untuk bucket

`get-bucket-inventory-configuration` Contoh berikut mengambil konfigurasi inventaris untuk bucket yang ditentukan dengan ID1.

```
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"
    }
  }
}
```

- Untuk detail API, lihat [GetBucketInventoryConfiguration](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengembalikan detail inventaris bernama 'testinventory' untuk bucket S3 yang diberikan.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId 'testinventory'
```

- Untuk detail API, lihat [GetBucketInventoryConfiguration](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengembalikan detail inventaris bernama 'testinventory' untuk bucket S3 yang diberikan.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId 'testinventory'
```

- Untuk detail API, lihat [GetBucketInventoryConfiguration](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketLifecycleConfiguration** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `GetBucketLifecycleConfiguration`.

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
}
```

- Untuk detail API, lihat [GetBucketLifecycleConfiguration](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Perintah berikut mengambil konfigurasi siklus hidup untuk bucket bernama: `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"
    }
]
}

```

- Untuk detail API, lihat [GetBucketLifecycleConfiguration](#) di Referensi AWS CLI Perintah.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
```

- Untuk detail API, lihat [GetBucketLifecycleConfiguration](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.

```

- Untuk detail API, lihat [GetBucketLifecycleConfiguration](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketLocation** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `GetBucketLocation`.

CLI

AWS CLI

Perintah berikut mengambil batasan lokasi untuk bucket bernama `amzn-s3-demo-bucket`, jika ada kendala:

```
aws s3api get-bucket-location --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "LocationConstraint": "us-west-2"
}
```

- Untuk detail API, lihat [GetBucketLocation](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengembalikan batasan lokasi untuk bucket 'amzn-s3-demo-bucket', jika ada kendala.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Value
-----
ap-south-1
```

- Untuk detail API, lihat [GetBucketLocation](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengembalikan batasan lokasi untuk bucket 'amzn-s3-demo-bucket', jika ada kendala.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Value
-----
ap-south-1
```

- Untuk detail API, lihat [GetBucketLocation](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Rust

SDK for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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(())
}
```

- Untuk detail API, lihat [GetBucketLocation](#) referensi AWS SDK for Rust API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketLogging** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan `GetBucketLogging`.

CLI

AWS CLI

Untuk mengambil status logging untuk bucket

`get-bucket-logging` Contoh berikut mengambil status logging untuk bucket yang ditentukan.

```
aws s3api get-bucket-logging \
    --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "LoggingEnabled": {
    "TargetPrefix": "",
    "TargetBucket": "amzn-s3-demo-bucket-logs"
  }
}
```

```
}
```

- Untuk detail API, lihat [GetBucketLogging](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengembalikan status logging untuk bucket yang ditentukan.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Output:

TargetBucketName	Grants	TargetPrefix
testbucket1	{}	testprefix

- Untuk detail API, lihat [GetBucketLogging](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengembalikan status logging untuk bucket yang ditentukan.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Output:

TargetBucketName	Grants	TargetPrefix
testbucket1	{}	testprefix

- Untuk detail API, lihat [GetBucketLogging](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketMetricsConfiguration** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan **GetBucketMetricsConfiguration**.

CLI

AWS CLI

Untuk mengambil konfigurasi metrik untuk bucket dengan ID tertentu

get-bucket-metrics-configuration Contoh berikut menampilkan konfigurasi metrik untuk bucket dan ID yang ditentukan.

```
aws s3api get-bucket-metrics-configuration \
  --bucket amzn-s3-demo-bucket \
  --id 123
```

Output:

```
{
  "MetricsConfiguration": {
    "Filter": {
      "Prefix": "logs"
    },
    "Id": "123"
  }
}
```

- Untuk detail API, lihat [GetBucketMetricsConfiguration](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengembalikan detail tentang filter metrik bernama 'testfilter' untuk bucket S3 yang diberikan.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId
'testfilter'
```

- Untuk detail API, lihat [GetBucketMetricsConfiguration](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengembalikan detail tentang filter metrik bernama 'testfilter' untuk bucket S3 yang diberikan.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId 'testfilter'
```

- Untuk detail API, lihat [GetBucketMetricsConfiguration](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketNotification** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan `GetBucketNotification`.

CLI

AWS CLI

Perintah berikut mengambil konfigurasi notifikasi untuk bucket bernama `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:*"
    ]
  }
}
```

- Untuk detail API, lihat [GetBucketNotification](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Contoh ini mengambil konfigurasi notifikasi dari bucket yang diberikan

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty  
TopicConfigurations
```

Output:

```
Id      Topic  
--      -  
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Untuk detail API, lihat [GetBucketNotification](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Contoh ini mengambil konfigurasi notifikasi dari bucket yang diberikan

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty  
TopicConfigurations
```

Output:

```
Id      Topic  
--      -  
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Untuk detail API, lihat [GetBucketNotification](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketPolicy** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `GetBucketPolicy`.

C++

SDK untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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();
}
```

```
}
```

- Untuk detail API, lihat [GetBucketPolicy](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Perintah berikut mengambil kebijakan bucket untuk bucket bernama `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "Policy": "{\"Version\":\"2008-10-17\",\"Statement\": [{\"Sid\":\"\", \"Effect\": \"Allow\", \"Principal\": \"*\", \"Action\": \"s3:GetObject\", \"Resource\": \"arn:aws:s3:::amzn-s3-demo-bucket/*\"}, {\"Sid\":\"\", \"Effect\": \"Deny\", \"Principal\": \"*\", \"Action\": \"s3:GetObject\", \"Resource\": \"arn:aws:s3:::amzn-s3-demo-bucket/secret/*\"}]}"
}
```

Dapatkan dan letakkan kebijakan bucket Contoh berikut menunjukkan bagaimana Anda dapat mengunduh kebijakan bucket Amazon S3, membuat modifikasi pada file, dan kemudian `put-bucket-policy` menggunakannya untuk menerapkan kebijakan bucket yang dimodifikasi. Untuk mengunduh kebijakan bucket ke file, Anda dapat menjalankan:

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket --query Policy --output text > policy.json
```

Anda kemudian dapat memodifikasi `policy.json` file sesuai kebutuhan. Terakhir, Anda dapat menerapkan kebijakan yang dimodifikasi ini kembali ke bucket S3 dengan menjalankan:

`policy.json` berkas sesuai kebutuhan. Terakhir, Anda dapat menerapkan kebijakan yang dimodifikasi ini kembali ke bucket S3 dengan menjalankan:

berkas sesuai kebutuhan. Terakhir, Anda dapat menerapkan kebijakan yang dimodifikasi ini kembali ke bucket S3 dengan menjalankan:

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://  
policy.json
```

- Untuk detail API, lihat [GetBucketPolicy](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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 "";  
    }  
}
```

- Untuk detail API, lihat [GetBucketPolicy](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Dapatkan kebijakan 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;
    }
}
};

```

- Untuk informasi selengkapnya, lihat [AWS SDK untuk JavaScript Panduan Developer](#).
- Untuk detail API, lihat [GetBucketPolicy](#) di Referensi AWS SDK untuk JavaScript API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
    }
}
```

- Untuk detail API, lihat [GetBucketPolicy](#) di AWS SDK untuk referensi API Kotlin.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini menampilkan kebijakan bucket yang terkait dengan bucket S3 yang diberikan.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [GetBucketPolicy](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini menampilkan kebijakan bucket yang terkait dengan bucket S3 yang diberikan.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [GetBucketPolicy](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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)

```

- Untuk detail API, lihat [GetBucketPolicy](#) di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

end
```

- Untuk detail API, lihat [GetBucketPolicy](#) di Referensi AWS SDK untuk Ruby API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [GetBucketPolicy](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketPolicyStatus** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan **GetBucketPolicyStatus**.

CLI

AWS CLI

Untuk mengambil status kebijakan untuk bucket yang menunjukkan apakah bucket bersifat publik

get-bucket-policy-status Contoh berikut mengambil status kebijakan untuk bucket **amzn-s3-demo-bucket**.

```
aws s3api get-bucket-policy-status \  
  --bucket amzn-s3-demo-bucket
```

Output:

```
{  
  "PolicyStatus": {  
    "IsPublic": false  
  }  
}
```

- Untuk detail API, lihat [GetBucketPolicyStatus](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengembalikan status kebijakan untuk bucket S3 yang diberikan, yang menunjukkan apakah bucket bersifat publik.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [GetBucketPolicyStatus](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengembalikan status kebijakan untuk bucket S3 yang diberikan, yang menunjukkan apakah bucket bersifat publik.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [GetBucketPolicyStatus](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketReplication** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `GetBucketReplication`.

CLI

AWS CLI

Perintah berikut mengambil konfigurasi replikasi untuk bucket bernama: `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"
  }
}
```

- Untuk detail API, lihat [GetBucketReplication](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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());
    }
}
```

- Untuk detail API, lihat [GetBucketReplication](#) di Referensi AWS SDK for Java 2.x API.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Mengembalikan informasi konfigurasi replikasi yang disetel pada bucket bernama 'amzn-s3-demo-bucket'.

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Untuk detail API, lihat [GetBucketReplication](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Mengembalikan informasi konfigurasi replikasi yang disetel pada bucket bernama 'amzn-s3-demo-bucket'.

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Untuk detail API, lihat [GetBucketReplication](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketRequestPayment** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan **GetBucketRequestPayment**.

CLI

AWS CLI

Untuk mengambil konfigurasi pembayaran permintaan untuk bucket

`get-bucket-request-payment` Contoh berikut mengambil konfigurasi requester pay untuk bucket yang ditentukan.

```
aws s3api get-bucket-request-payment \  
  --bucket amzn-s3-demo-bucket
```

Output:

```
{  
  "Payer": "BucketOwner"  
}
```

- Untuk detail API, lihat [GetBucketRequestPayment](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Mengembalikan konfigurasi pembayaran permintaan untuk bucket bernama 'amzn-s3-demo-bucket'. Secara default, pemilik bucket membayar unduhan dari bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Untuk detail API, lihat [GetBucketRequestPayment](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Mengembalikan konfigurasi pembayaran permintaan untuk bucket bernama 'amzn-s3-demo-bucket'. Secara default, pemilik bucket membayar unduhan dari bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Untuk detail API, lihat [GetBucketRequestPayment](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketTagging** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan `GetBucketTagging`.

CLI

AWS CLI

Perintah berikut mengambil konfigurasi penandaan untuk bucket bernama: `amzn-s3-demo-bucket`

```
aws s3api get-bucket-tagging --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "TagSet": [
    {
      "Value": "marketing",
      "Key": "organization"
    }
  ]
}
```

```
}  
]  
}
```

- Untuk detail API, lihat [GetBucketTagging](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengembalikan semua tag yang terkait dengan bucket yang diberikan.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [GetBucketTagging](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengembalikan semua tag yang terkait dengan bucket yang diberikan.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [GetBucketTagging](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketVersioning** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan `GetBucketVersioning`.

CLI

AWS CLI

Perintah berikut mengambil konfigurasi pembuatan versi untuk bucket bernama: `amzn-s3-demo-bucket`

```
aws s3api get-bucket-versioning --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "Status": "Enabled"
}
```

- Untuk detail API, lihat [GetBucketVersioning](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengembalikan status pembuatan versi sehubungan dengan bucket yang diberikan.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [GetBucketVersioning](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengembalikan status pembuatan versi sehubungan dengan bucket yang diberikan.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [GetBucketVersioning](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketWebsite** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `GetBucketWebsite`.

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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}");
```

- Untuk detail API, lihat [GetBucketWebsite](#) di Referensi AWS SDK untuk .NET API.

C++

SDK untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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();
}

```

- Untuk detail API, lihat [GetBucketWebsite](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Perintah berikut mengambil konfigurasi situs web statis untuk bucket bernama `amzn-s3-demo-bucket`:


```
aws s3api get-bucket-website --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "IndexDocument": {
    "Suffix": "index.html"
  },
  "ErrorDocument": {
    "Key": "error.html"
  }
}
```

- Untuk detail API, lihat [GetBucketWebsite](#) di Referensi AWS CLI Perintah.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Dapatkan konfigurasi situs 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;
  }
}
};
```

- Untuk detail API, lihat [GetBucketWebsite](#) di Referensi AWS SDK untuk JavaScript API.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengembalikan detail konfigurasi situs web statis dari bucket S3 yang diberikan.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [GetBucketWebsite](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengembalikan detail konfigurasi situs web statis dari bucket S3 yang diberikan.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [GetBucketWebsite](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetObject** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `GetObject`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)
- [Mendapatkan objek dari bucket jika telah diubah](#)
- [Dapatkan objek dari Multi-Region Access Point](#)
- [Memulai enkripsi](#)
- [Memulai dengan penyimpanan objek](#)
- [Buat permintaan bersyarat](#)
- [Lacak unggahan dan unduhan](#)

.NET

SDK untuk .NET (v4)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
    }
}

```

- Untuk detail API, lihat [GetObject](#) di Referensi AWS SDK untuk .NET API.

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Dapatkan objek menggunakan permintaan bersyarat.

```

/// <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;
    }
}

```

- Untuk detail API, lihat [GetObject](#) di Referensi AWS SDK untuk .NET API.

Bash

AWS CLI dengan skrip Bash

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat [GetObject](#) di Referensi AWS CLI Perintah.

C++

SDK untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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();
}

```

- Untuk detail API, lihat [GetObject](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Contoh berikut menggunakan `get-object` perintah untuk mengunduh objek dari Amazon S3:

```
aws s3api get-object --bucket text-content --key dir/
my_images.tar.bz2 my_images.tar.bz2
```

Perhatikan bahwa parameter outfile ditentukan tanpa nama opsi seperti “--outfile”. Nama file output harus menjadi parameter terakhir dalam perintah.

Contoh di bawah ini menunjukkan penggunaan `--range` untuk men-download rentang byte tertentu dari sebuah objek. Perhatikan rentang byte perlu diawali dengan “byte =”:

```
aws s3api get-object --bucket text-content --key dir/my_data --
range bytes=8888-9999 my_data_range
```

Untuk informasi selengkapnya tentang mengambil objek, lihat Mendapatkan Objek di Panduan Pengembang Amazon S3.

- Untuk detail API, lihat [GetObject](#) di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat [GetObject](#) di Referensi AWS SDK untuk Go API.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Membaca data sebagai array byte menggunakan [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);
        }
    });
}
```

Gunakan file [S3TransferManager](#) untuk [mengunduh objek](#) dalam bucket S3 ke file lokal. Lihat [file lengkap](#) dan [lakukan pengujian](#).

```
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().contentType());
        return downloadResult.response().contentType();
    }

```

Baca tanda milik objek menggunakan [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.tagSet();
        for (Tag tag : tagSet) {
            System.out.println(tag.key());
            System.out.println(tag.value());
        }

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

Dapatkan URL untuk objek menggunakan [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);
    }
}
}

```

Dapatkan objek dengan menggunakan objek klien S3Presigner menggunakan [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.setRequestProperty(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();
}

}
}

```

Dapatkan objek dengan menggunakan ResponseTransformer objek dan [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);
        }
    }
}

```

- Untuk detail API, lihat [GetObject](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Unduh objek tersebut.

```
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;
    }
}
};

```

Unduh objek dengan syarat ETag cocok dengan yang disediakan.

```

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"));
}
}
```

Unduh objek dengan syarat ETag tidak cocok dengan yang disediakan.

```
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"));
  }
}
```

```
}  
}
```

Unduh objek menggunakan dengan syarat telah dibuat atau dimodifikasi dalam jangka waktu tertentu.

```
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"));  
    }  
}
```

Unduh objek menggunakan dengan syarat belum dibuat atau dimodifikasi dalam jangka waktu tertentu.

```
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"));
    }
}
```

- Untuk informasi selengkapnya, silakan lihat [Panduan Developer AWS SDK untuk JavaScript](#).
- Untuk detail API, lihat [GetObject](#) di Referensi AWS SDK untuk JavaScript API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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")  
        }  
    }  
}
```

- Untuk detail API, lihat [GetObject](#) di AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Dapatkan objek.

```
$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.");
}
```

- Untuk detail API, lihat [GetObject](#) di Referensi AWS SDK untuk PHP API.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengambil item "sample.txt" dari bucket "amzn-s3-demo-bucket" dan menyimpannya ke file bernama "local-sample.txt" di lokasi saat ini. File "local-sample.txt" tidak harus ada sebelum perintah ini dipanggil.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-
sample.txt
```

Contoh 2: Perintah ini mengambil direktori virtual “DIR” dari bucket “amzn-s3-demo-bucket” dan menyimpannya ke folder bernama “” di lokasi saat ini. Local-DIR Folder Local-DIR “” tidak harus ada sebelum perintah ini dipanggil.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Contoh 3: Mengunduh semua objek dengan kunci yang diakhiri dengan '.json' dari ember dengan 'konfigurasi' dalam nama ember ke file di folder yang ditentukan. Kunci objek digunakan untuk mengatur nama file.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\Config0bjects
```

- Untuk detail API, lihat [GetObject](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengambil item "sample.txt" dari bucket “amzn-s3-demo-bucket” dan menyimpannya ke file bernama "local-sample.txt" di lokasi saat ini. File "local-sample.txt" tidak harus ada sebelum perintah ini dipanggil.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

Contoh 2: Perintah ini mengambil direktori virtual “DIR” dari bucket “amzn-s3-demo-bucket” dan menyimpannya ke folder bernama “” di lokasi saat ini. Local-DIR Folder Local-DIR “” tidak harus ada sebelum perintah ini dipanggil.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Contoh 3: Mengunduh semua objek dengan kunci yang diakhiri dengan '.json' dari ember dengan 'konfigurasi' dalam nama ember ke file di folder yang ditentukan. Kunci objek digunakan untuk mengatur nama file.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\Config0bjects
```

Contoh 4: Mengunduh satu objek menggunakan unduhan paralel multibagian untuk meningkatkan kinerja pada file besar.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt -UseMultipartDownload
```

Contoh 5: Mengunduh objek besar menggunakan mode RANGE dengan ukuran bagian 64MB khusus. Mode RANGE menggunakan permintaan rentang byte HTTP dan berfungsi untuk semua objek terlepas dari bagaimana mereka diunggah.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -PartSize 64MB
```

Contoh 6: Mengunduh semua objek dengan key prefix “data/” ke folder lokal menggunakan unduhan multipart dengan unduhan file bersamaan.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-data -UseMultipartDownload -DownloadFilesConcurrently
```

Contoh 7: Mengunduh satu objek menggunakan unduhan multibagian dengan 20 koneksi bersamaan untuk throughput maksimum pada jaringan bandwidth tinggi.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

Contoh 8: Mengunduh semua objek dengan key prefix “data/” ke folder lokal menggunakan unduhan multipart, melanjutkan kegagalan file individual alih-alih membatalkan.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- Untuk detail API, lihat [GetObject](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
```

Dapatkan objek menggunakan permintaan bersyarat.

```
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

```

- Untuk detail API, lihat [GetObject](#) di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Dapatkan objek.

```

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__

```

Dapatkan objek dan laporkan status enkripsi di sisi servernya.

```

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__
```

- Untuk detail API, lihat [GetObject](#) di Referensi AWS SDK untuk Ruby API.

Rust

SDK for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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)
}
```

- Untuk detail API, lihat [GetObject](#) referensi AWS SDK for Rust API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [GetObject](#) di AWS SDK untuk referensi SAP ABAP API.

Swift

SDK para Swift

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
    }
}
```

- Untuk detail API, lihat referensi [GetObject AWSSDK](#) untuk Swift API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetObjectAcl** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `GetObjectAcl`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Mengelola daftar kontrol akses \(ACL\)](#)

C++

SDK untuk C++

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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";
        }
    }
}

```

- Untuk detail API, lihat [GetObjectAcl](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Perintah berikut mengambil daftar kontrol akses untuk objek dalam bucket bernama `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"
    }
  ]
}
```

- Untuk detail API, lihat [GetObjectAcl](#) di Referensi AWS CLI Perintah.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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}")
        }
    }
}
```

- Untuk detail API, lihat [GetObjectAcl](#) di AWS SDK untuk referensi API Kotlin.

PowerShell

Alat untuk PowerShell V5

Contoh 1: Perintah mendapatkan rincian pemilik objek objek S3.

```
(Get-S3ObjectACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select
*).Owner
```

Output:

```

DisplayName Id
-----
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544

```

- Untuk detail API, lihat [GetObjectAcl](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

```

- Untuk detail API, lihat [GetObjectAcl](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.

```

- Untuk detail API, lihat [GetObjectAcl](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetObjectAttributes** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `GetObjectAttributes`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Bekerja dengan integritas objek Amazon S3](#)

C++

SDK untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
}

```

- Untuk detail API, lihat [GetObjectAttributes](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Untuk mengambil metadata dari objek tanpa mengembalikan objek itu sendiri

get-object-attributes Contoh berikut mengambil metadata dari objek. 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
}
```

Untuk informasi selengkapnya, lihat [GetObjectAttributes](#) di Referensi API Amazon S3.

- Untuk detail API, lihat [GetObjectAttributes](#) di Referensi AWS CLI Perintah.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetObjectLegalHold** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `GetObjectLegalHold`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Kunci objek Amazon S3](#)

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh selengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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();
    }
}

```

- Untuk detail API, lihat [GetObjectLegalHold](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Mengambil status Legal Hold dari suatu objek

get-object-legal-hold Contoh berikut mengambil status Penahanan Hukum untuk objek yang ditentukan.

```

aws s3api get-object-legal-hold \
  --bucket amzn-s3-demo-bucket-with-object-lock \
  --key doc1.rtf

```

Output:

```

{
  "LegalHold": {
    "Status": "ON"
  }
}

```

```
}
```

- Untuk detail API, lihat [GetObjectLegalHold](#) di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat [GetObjectLegalHold](#) di Referensi AWS SDK untuk Go API.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
}
```

- Untuk detail API, lihat [GetObjectLegalHold](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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"));
  }
}
```

- Untuk detail API, lihat [GetObjectLegalHold](#) di Referensi AWS SDK untuk JavaScript API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Letakkan benda penahanan hukum.

```
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,
        )
```

- Untuk detail API, lihat [GetObjectLegalHold](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [GetObjectLegalHold](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetObjectLockConfiguration** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `GetObjectLockConfiguration`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Kunci objek Amazon S3](#)

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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();
    }
}
```

```
}
```

- Untuk detail API, lihat [GetObjectLockConfiguration](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Untuk mengambil konfigurasi kunci objek untuk bucket

`get-object-lock-configuration` Contoh berikut mengambil konfigurasi kunci objek untuk bucket yang ditentukan.

```
aws s3api get-object-lock-configuration \  
  --bucket amzn-s3-demo-bucket-with-object-lock
```

Output:

```
{  
  "ObjectLockConfiguration": {  
    "ObjectLockEnabled": "Enabled",  
    "Rule": {  
      "DefaultRetention": {  
        "Mode": "COMPLIANCE",  
        "Days": 50  
      }  
    }  
  }  
}
```

- Untuk detail API, lihat [GetObjectLockConfiguration](#) di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat [GetObjectLockConfiguration](#) di Referensi AWS SDK untuk Go API.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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());
}

```

- Untuk detail API, lihat [GetObjectLockConfiguration](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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"));  
  }  
}
```

- Untuk detail API, lihat [GetObjectLockConfiguration](#) di Referensi AWS SDK untuk JavaScript API.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengembalikan nilai 'Diaktifkan' jika konfigurasi kunci Objek diaktifkan untuk bucket S3 yang diberikan.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select  
ObjectLockConfiguration.ObjectLockEnabled
```

Output:

```
Value  
-----  
Enabled
```

- Untuk detail API, lihat [GetObjectLockConfiguration](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengembalikan nilai 'Diaktifkan' jika konfigurasi kunci Objek diaktifkan untuk bucket S3 yang diberikan.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select  
ObjectLockConfiguration.ObjectLockEnabled
```

Output:

```
Value
-----
Enabled
```

- Untuk detail API, lihat [GetObjectLockConfiguration](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Dapatkan konfigurasi kunci objek.

```
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
```

- Untuk detail API, lihat [GetObjectLockConfiguration](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [GetObjectLockConfiguration](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetObjectRetention** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `GetObjectRetention`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Kunci objek Amazon S3](#)

.NET

SDK untuk .NET

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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($"Unable to fetch object lock retention:
'{ex.Message}');
        return new ObjectLockRetention();
    }
}
```

- Untuk detail API, lihat [GetObjectRetention](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Untuk mengambil konfigurasi retensi objek untuk objek

`get-object-retention` Contoh berikut mengambil konfigurasi retensi objek untuk objek tertentu.

```
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"
  }
}
```

- Untuk detail API, lihat [GetObjectRetention](#) di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat [GetObjectRetention](#) di Referensi AWS SDK untuk Go API.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
    }
}

```

- Untuk detail API, lihat [GetObjectRetention](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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"));
  }
}
```

- Untuk detail API, lihat [GetObjectRetention](#) di Referensi AWS SDK untuk JavaScript API.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah mengembalikan mode dan tanggal sampai objek akan dipertahankan.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- Untuk detail API, lihat [GetObjectRetention](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah mengembalikan mode dan tanggal sampai objek akan dipertahankan.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```


- Untuk detail API, lihat [GetObjectRetention](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetObjectTagging** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan `GetObjectTagging`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Memulai dengan tanda](#)

CLI

AWS CLI

Untuk mengambil tag yang dilampirkan ke objek

`get-object-tagging` Contoh berikut mengambil nilai-nilai untuk kunci tertentu dari objek tertentu.

```
aws s3api get-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc1.rtf
```

Output:

```
{  
  "TagSet": [  
    {  
      "Value": "confidential",  
      "Key": "designation"  
    }  
  ]  
}
```

get-object-tagging Contoh berikut mencoba untuk mengambil set tag objek `doc2.rtf`, yang tidak memiliki tag.

```
aws s3api get-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc2.rtf
```

Output:

```
{  
  "TagSet": []  
}
```

get-object-tagging Contoh berikut mengambil set tag objek `doc3.rtf`, yang memiliki beberapa 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"  
    }  
  ]  
}
```

- Untuk detail API, lihat [GetObjectTagging](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Sampel mengembalikan tag yang terkait dengan objek yang ada pada bucket S3 yang diberikan.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Key  Value
---  -
test value
```

- Untuk detail API, lihat [GetObjectTagging](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Sampel mengembalikan tag yang terkait dengan objek yang ada pada bucket S3 yang diberikan.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Output:

```
Key  Value
---  -
test value
```

- Untuk detail API, lihat [GetObjectTagging](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetPublicAccessBlock** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan `GetPublicAccessBlock`.

CLI

AWS CLI

Untuk menyetel atau memodifikasi konfigurasi blokir akses publik untuk bucket

`get-public-access-block` Contoh berikut menampilkan konfigurasi blokir akses publik untuk bucket yang ditentukan.

```
aws s3api get-public-access-block \
  --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "PublicAccessBlockConfiguration": {
    "IgnorePublicAcls": true,
    "BlockPublicPolicy": true,
    "BlockPublicAcls": true,
    "RestrictPublicBuckets": true
  }
}
```

- Untuk detail API, lihat [GetPublicAccessBlock](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah mengembalikan konfigurasi blok akses publik dari bucket S3 yang diberikan.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [GetPublicAccessBlock](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah mengembalikan konfigurasi blok akses publik dari bucket S3 yang diberikan.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [GetPublicAccessBlock](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **HeadBucket** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan **HeadBucket**.

Bash

AWS CLI dengan skrip Bash

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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  
}
```

- Untuk detail API, lihat [HeadBucket](#) di Referensi AWS CLI Perintah.

CLI

AWS CLI

Perintah berikut memverifikasi akses ke bucket bernama `amzn-s3-demo-bucket`:

```
aws s3api head-bucket --bucket amzn-s3-demo-bucket
```

Jika bucket ada dan Anda memiliki akses ke sana, tidak ada output yang dikembalikan. Jika tidak, pesan kesalahan akan ditampilkan. Contoh:

```
A client error (404) occurred when calling the HeadBucket operation: Not Found
```

- Untuk detail API, lihat [HeadBucket](#) di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat [HeadBucket](#) di Referensi AWS SDK untuk Go API.

PowerShell

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengembalikan output dengan kode status HTTP 200 OK untuk bucket yang ada saat pengguna memiliki izin untuk mengaksesnya. BucketArn parameter hanya didukung untuk bucket direktori 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
```

Contoh 2: Perintah ini menampilkan kesalahan dengan kode status HTTP NotFound untuk bucket yang tidak ada.

```
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.
```

Contoh 3: Perintah ini menampilkan kesalahan dengan kode status HTTP Forbidden untuk bucket yang ada di mana pengguna tidak memiliki izin untuk mengaksesnya.

```
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.
```

- Untuk detail API, lihat [HeadBucket](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Python

SDK untuk Python (Boto3)

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
```

- Untuk detail API, lihat [HeadBucket](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [HeadBucket](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **HeadObject** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `HeadObject`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Periksa apakah ada objek](#)
- [Memulai dengan penyimpanan objek](#)

CLI

AWS CLI

Perintah berikut mengambil metadata untuk objek dalam bucket bernama: `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": {}
}
```

- Untuk detail API, lihat [HeadObject](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Tentukan jenis konten suatu objek.

```
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);
    }
}
}
```

Dapatkan status pemulihan suatu objek.

```
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);
        }
    }
}

```

- Untuk detail API, lihat [HeadObject](#) di Referensi AWS SDK for Java 2.x API.

Ruby

SDK untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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__

```

- Untuk detail API, lihat [HeadObject](#) di Referensi AWS SDK untuk Ruby API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **ListBucketAnalyticsConfigurations** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan **ListBucketAnalyticsConfigurations**.

CLI

AWS CLI

Untuk mengambil daftar konfigurasi analitik untuk bucket

Berikut ini akan `list-bucket-analytics-configurations` mengambil daftar konfigurasi analitik untuk bucket yang ditentukan.

```
aws s3api list-bucket-analytics-configurations \
  --bucket amzn-s3-demo-bucket
```

Output:

```
{
  "AnalyticsConfigurationList": [
    {
      "StorageClassAnalysis": {},
      "Id": "1"
    }
  ],
  "IsTruncated": false
}
```

- Untuk detail API, lihat [ListBucketAnalyticsConfigurations](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengembalikan 100 konfigurasi analitik pertama dari bucket S3 yang diberikan.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [ListBucketAnalyticsConfigurations](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengembalikan 100 konfigurasi analitik pertama dari bucket S3 yang diberikan.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [ListBucketAnalyticsConfigurations](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **ListBucketInventoryConfigurations** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan **ListBucketInventoryConfigurations**.

CLI

AWS CLI

Untuk mengambil daftar konfigurasi inventaris untuk bucket

list-bucket-inventory-configurations Contoh berikut mencantumkan konfigurasi inventaris untuk bucket yang ditentukan.

```
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
}

```

- Untuk detail API, lihat [ListBucketInventoryConfigurations](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengembalikan 100 konfigurasi inventaris pertama dari bucket S3 yang diberikan.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [ListBucketInventoryConfigurations](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengembalikan 100 konfigurasi inventaris pertama dari bucket S3 yang diberikan.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Untuk detail API, lihat [ListBucketInventoryConfigurations](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **ListBuckets** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `ListBuckets`.

.NET

SDK untuk .NET

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
    }
}
}

```

- Untuk detail API, lihat [ListBuckets](#) di Referensi AWS SDK untuk .NET API.

C++

SDK untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
}
```

- Untuk detail API, lihat [ListBuckets](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Perintah berikut menggunakan `list-buckets` perintah untuk menampilkan nama semua bucket Amazon S3 Anda (di semua wilayah):

```
aws s3api list-buckets --query "Buckets[].Name"
```

Opsi kueri menyaring output dari `list-buckets` down ke hanya nama bucket.

Untuk informasi selengkapnya tentang bucket, lihat [Bekerja dengan Bucket Amazon S3](#) di [Panduan Pengembang Amazon S3](#).

- Untuk detail API, lihat [ListBuckets](#) di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat [ListBuckets](#) di Referensi AWS SDK untuk Go API.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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());  
    }
```

- Untuk detail API, lihat [ListBuckets](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat daftar 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;
  }
}
};
```

- Untuk informasi selengkapnya, lihat [AWS SDK untuk JavaScript Panduan Developer](#).
- Untuk detail API, lihat [ListBuckets](#) di Referensi AWS SDK untuk JavaScript API.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengembalikan semua bucket S3.

```
Get-S3Bucket
```

Contoh 2: Perintah ini mengembalikan bucket bernama “test-files”

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Untuk detail API, lihat [ListBuckets](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengembalikan semua bucket S3.

```
Get-S3Bucket
```

Contoh 2: Perintah ini mengembalikan bucket bernama “test-files”

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Untuk detail API, lihat [ListBuckets](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

```

- Untuk detail API, lihat [ListBuckets](#) di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

  # 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

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__

```

- Untuk detail API, lihat [ListBuckets](#) di Referensi AWS SDK untuk Ruby API.

Rust

SDK for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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(())
}

```

- Untuk detail API, lihat [ListBuckets](#) referensi AWS SDK for Rust API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [ListBuckets](#) di AWS SDK untuk referensi SAP ABAP API.

Swift

SDK para Swift

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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())  
  }
```

- Untuk detail API, lihat referensi [ListBuckets AWS SDK](#) untuk Swift API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **ListMultipartUploads** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `ListMultipartUploads`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Hapus unggahan multipart yang tidak lengkap](#)

CLI

AWS CLI

Perintah berikut mencantumkan semua unggahan multipart aktif untuk bucket bernama: `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":  
        "dfRtDYU0WWCCcH43C3WfbkR0NycyCpTJJvxu2i5GYkZlJf.Yxwh6XG7WfS2vC4to6HiV6Yj1x.cph0gtNBtJ8P3"
```

```

        "StorageClass": "STANDARD",
        "Key": "multipart/01",
        "Owner": {
            "DisplayName": "aws-account-name",
            "ID":
"100719349fc3b6dcd7c820a124bf7aec408092c3d7b51b38494939801fc248b"
        }
    },
    "CommonPrefixes": []
}

```

Unggahan multipart yang sedang berlangsung menimbulkan biaya penyimpanan di Amazon S3. Selesaikan atau batalkan unggahan multibagian aktif untuk menghapus bagian-bagiannya dari akun Anda.

- Untuk detail API, lihat [ListMultipartUploads](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
    }
}
```

- Untuk detail API, lihat [ListMultipartUploads](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **ListObjectVersions** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `ListObjectVersions`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Memulai dengan penyimpanan objek](#)
- [Bekerja dengan objek berversi](#)

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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}'");
    }
}
}

```

- Untuk detail API, lihat [ListObjectVersions](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Perintah berikut mengambil informasi versi untuk objek dalam bucket bernama `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_l2T8UHDkFEwCgJjhlGp0ZC0qJ.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
}
]
}

```

- Untuk detail API, lihat [ListObjectVersions](#) di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat [ListObjectVersions](#) di Referensi AWS SDK untuk Go API.

Rust

SDK for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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(())
}

```

- Untuk detail API, lihat [ListObjectVersions](#) referensi AWS SDK for Rust API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.

```

- Untuk detail API, lihat [ListObjectVersions](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **ListObjects** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan `ListObjects`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Membuat halaman web yang mencantumkan objek Amazon S3](#)

CLI

AWS CLI

Contoh berikut menggunakan `list-objects` perintah untuk menampilkan nama semua objek dalam bucket yang ditentukan:

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size: Size}'
```

Contoh menggunakan `--query` argumen untuk memfilter output `list-objects` turun ke nilai kunci dan ukuran untuk setiap objek

Untuk informasi selengkapnya tentang objek, lihat [Bekerja dengan Objek Amazon S3](#) di [Panduan Pengembang Amazon S3](#).

- Untuk detail API, lihat [ListObjects](#) di [Referensi AWS CLI Perintah](#).

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengambil informasi tentang semua item di bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Contoh 2: Perintah ini mengambil informasi tentang item "sample.txt" dari bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Contoh 3: Perintah ini mengambil informasi tentang semua item dengan awalan “sample” dari bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Untuk detail API, lihat [ListObjects](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengambil informasi tentang semua item di bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Contoh 2: Perintah ini mengambil informasi tentang item "sample.txt" dari bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Contoh 3: Perintah ini mengambil informasi tentang semua item dengan awalan “sample” dari bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Untuk detail API, lihat [ListObjects](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **ListObjectsV2** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `ListObjectsV2`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)
- [Hapus semua objek dalam ember](#)
- [Memulai dengan penyimpanan objek](#)

.NET

SDK untuk .NET (v4)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
    }
}
```

- Untuk detail API, lihat [ListObjectsV2](#) di Referensi AWS SDK untuk .NET API.

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Daftar objek dengan paginator.

```
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}");
            }
        }
    }
}
```



```
}
}
```

- Untuk detail API, lihat [ListObjectsV2](#) di Referensi AWS SDK untuk .NET API.

Bash

AWS CLI dengan skrip Bash

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat [ListObjectsV2](#) di Referensi AWS CLI Perintah.

C++

SDK untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
}
```

- Untuk detail API, lihat [ListObjectsV2](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Untuk mendapatkan daftar objek dalam ember

`list-objects-v2` Contoh berikut mencantumkan objek dalam bucket yang ditentukan.

```
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
    }
  ]
}
```

- Untuk detail API, lihat [ListObjectsV2](#) di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat [ListObjectsV2](#) di Referensi AWS SDK untuk Go API.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
        });
    }
}

```

Buat daftar objek menggunakan penomoran halaman.

```

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);
    }
}
}
```


- Untuk detail API, lihat [ListObjectsV2](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat daftar semua objek di bucket Anda. Jika ada lebih dari satu objek, `IsTruncated` dan `NextContinuationToken` akan digunakan untuk iterasi atas daftar lengkap.

```
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;
    }
}
};
```

- Untuk detail API, lihat [ListObjectsV2](#) di Referensi AWS SDK untuk JavaScript API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

```

- Untuk detail API, lihat [ListObjectsV2](#) di AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat daftar objek dalam 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.");
    }

```

- Untuk detail API, lihat [ListObjectsV2](#) di Referensi AWS SDK untuk PHP API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
```

- Untuk detail API, lihat [ListObjectsV2](#) di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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__

```

- Untuk detail API, lihat [ListObjectsV2](#) di Referensi AWS SDK untuk Ruby API.

Rust

SDK for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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(())
}
```

- Untuk detail API, lihat [ListObjectsV2](#) di AWS SDK untuk referensi Rust API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.

```

- Untuk detail API, lihat [ListObjectsV2](#) di AWS SDK untuk referensi SAP ABAP API.

Swift

SDK para Swift

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
}

```

- Untuk detail API, lihat [ListObjectsV2](#) di AWS SDK untuk referensi Swift API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutBucketAccelerateConfiguration** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan **PutBucketAccelerateConfiguration**.

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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>
    /// <param name="bucketName">The name of the Amazon S3 bucket for which
the
    /// 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");
    }
}
}
```

- Untuk detail API, lihat [PutBucketAccelerateConfiguration](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Untuk mengatur konfigurasi percepatan ember

`put-bucket-accelerate-configuration` Contoh berikut memungkinkan konfigurasi percepatan untuk bucket yang ditentukan.

```
aws s3api put-bucket-accelerate-configuration \
  --bucket amzn-s3-demo-bucket \
  --accelerate-configuration Status=Enabled
```

Perintah ini tidak menghasilkan output.

- Untuk detail API, lihat [PutBucketAccelerateConfiguration](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini memungkinkan percepatan transfer untuk bucket S3 yang diberikan.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- Untuk detail API, lihat [PutBucketAccelerateConfiguration](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini memungkinkan percepatan transfer untuk bucket S3 yang diberikan.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- Untuk detail API, lihat [PutBucketAccelerateConfiguration](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutBucketAcl** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan **PutBucketAcl**.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Mengelola daftar kontrol akses \(ACL\)](#)

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
    }

```

- Untuk detail API, lihat [PutBucketAcl](#) di Referensi AWS SDK untuk .NET API.

C++

SDK untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
}

```

- Untuk detail API, lihat [PutBucketAcl](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Contoh ini memberikan `full control` kepada dua AWS pengguna (`user1@example.com` dan `user2@example.com`) dan `read` izin untuk semua orang:


```
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
```

Lihat <http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html> detail tentang ACL kustom (perintah s3api ACL, seperti `put-bucket-acl`, menggunakan notasi argumen singkatan yang sama).

- Untuk detail API, lihat [PutBucketAcl](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
}
}
```

- Untuk detail API, lihat [PutBucketAcl](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Tempatkan bucket ACL.

```

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;
  }
}
};
```

- Untuk informasi selengkapnya, lihat [AWS SDK untuk JavaScript Panduan Developer](#).
- Untuk detail API, lihat [PutBucketAcl](#) di Referensi AWS SDK untuk JavaScript API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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 = ownerOb
            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")
    }
}

```

- Untuk detail API, lihat [PutBucketAcl](#) di AWS SDK untuk referensi API Kotlin.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

```

- Untuk detail API, lihat [PutBucketAcl](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.

```

- Untuk detail API, lihat [PutBucketAcl](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutBucketCors** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan **PutBucketCors**.

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
}
```

- Untuk detail API, lihat [PutBucketCors](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Contoh berikut memungkinkan PUT, POST, dan DELETE permintaan dari `www.example.com`, dan memungkinkan GET permintaan dari domain apa pun:

```
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
    }
  ]
}
```

- Untuk detail API, lihat [PutBucketCors](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
    }
}
```

- Untuk detail API, lihat [PutBucketCors](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Tambahkan aturan 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.
            }
          ]
        }
      })
    );
  } catch (error) {
    if (error instanceof S3ServiceException) {
      console.error(error.message);
    } else {
      console.error(error);
    }
  }
}
```

```

        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;
    }
}
};

```

- Untuk informasi selengkapnya, lihat [AWS SDK untuk JavaScript Panduan Developer](#).
- Untuk detail API, lihat [PutBucketCors](#) di Referensi AWS SDK untuk JavaScript API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
```

- Untuk detail API, lihat [PutBucketCors](#) di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
          }
        ]
      }
    )
  end
end
```

```

    }
  )
  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
end

```

- Untuk detail API, lihat [PutBucketCors](#) di Referensi AWS SDK untuk Ruby API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.

```

- Untuk detail API, lihat [PutBucketCors](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutBucketEncryption** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan **PutBucketEncryption**.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Memulai dengan penyimpanan objek](#)

.NET

SDK untuk .NET

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
}

```

- Untuk detail API, lihat [PutBucketEncryption](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Untuk mengonfigurasi enkripsi sisi server untuk bucket

put-bucket-encryptionContoh berikut menetapkan enkripsi AES256 sebagai default untuk bucket yang ditentukan.

```
aws s3api put-bucket-encryption \  
  --bucket amzn-s3-demo-bucket \  
  --server-side-encryption-configuration '{"Rules":  
  [{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}]}'
```

Perintah ini tidak menghasilkan output.

- Untuk detail API, lihat [PutBucketEncryption](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengaktifkan enkripsi sisi server AES256 default dengan Amazon S3 Managed Keys SSE-S3 () pada bucket yang diberikan.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
  @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
  ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Untuk detail API, lihat [PutBucketEncryption](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengaktifkan enkripsi sisi server AES256 default dengan Amazon S3 Managed Keys SSE-S3 () pada bucket yang diberikan.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
  @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
  ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Untuk detail API, lihat [PutBucketEncryption](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutBucketLifecycleConfiguration** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan **PutBucketLifecycleConfiguration**.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Hapus unggahan multipart yang tidak lengkap](#)
- [Kelola pesan besar menggunakan S3](#)
- [Bekerja dengan objek berversi](#)

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
}
```

```
}
```

- Untuk detail API, lihat [PutBucketLifecycleConfiguration](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Perintah berikut menerapkan konfigurasi siklus hidup ke bucket bernama: `amzn-s3-demo-bucket`

```
aws s3api put-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --  
lifecycle-configuration file://lifecycle.json
```

File `lifecycle.json` adalah dokumen JSON di folder saat ini yang menentukan dua aturan:

```
{  
  "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"  ]  
}
```



```
    }  
  ]  
}
```

Aturan pertama memindahkan file dengan awalan `rotated` ke Glacier pada tanggal yang ditentukan. Aturan kedua memindahkan versi objek lama ke Glacier ketika mereka tidak lagi saat ini. Untuk informasi tentang format stempel waktu yang dapat diterima, lihat [Menentukan Nilai Parameter dalam Panduan Pengguna CLI AWS](#).

- Untuk detail API, lihat [PutBucketLifecycleConfiguration](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
    }
}
}
```

- Untuk detail API, lihat [PutBucketLifecycleConfiguration](#) di Referensi AWS SDK for Java 2.x API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

```

- Untuk detail API, lihat [PutBucketLifecycleConfiguration](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.

```

- Untuk detail API, lihat [PutBucketLifecycleConfiguration](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutBucketLogging** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan **PutBucketLogging**.

.NET

SDK untuk .NET

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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();
    }
}

```

- Untuk detail API, lihat [PutBucketLogging](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Contoh 1: Untuk mengatur pencatatan kebijakan bucket

put-bucket-logging Contoh berikut menetapkan kebijakan logging untuk amzn-s3-demo-bucket. Pertama, berikan izin utama layanan logging dalam kebijakan bucket Anda menggunakan put-bucket-policy perintah.

```

aws s3api put-bucket-policy \
  --bucket amzn-s3-demo-bucket \
  --policy file://policy.json

```

Isi dari 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"}
      }
    }
  ]
}

```

Untuk menerapkan kebijakan pencatatan, gunakan `put-bucket-logging`.

```

aws s3api put-bucket-logging \
  --bucket amzn-s3-demo-bucket \
  --bucket-logging-status file://logging.json

```

Isi dari `logging.json`:

```

{
  "LoggingEnabled": {
    "TargetBucket": "amzn-s3-demo-bucket",
    "TargetPrefix": "Logs/"
  }
}

```

`put-bucket-policy` Perintah diperlukan untuk memberikan `s3:PutObject` izin kepada prinsipal layanan logging.

Untuk informasi selengkapnya, lihat [Logging Akses Server Amazon S3](#) di Panduan Pengguna Amazon S3.

Contoh 2: Untuk menetapkan kebijakan bucket untuk akses log hanya ke satu pengguna

`put-bucket-logging` Contoh berikut menetapkan kebijakan logging untuk `amzn-s3-demo-bucket`. AWS Pengguna `bob@example.com` akan memiliki kontrol penuh atas file log, dan tidak ada orang lain yang memiliki akses. Pertama, berikan izin S3 dengan `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

```

Kemudian terapkan kebijakan logging menggunakan `put-bucket-logging`.

```

aws s3api put-bucket-logging \

```

```
--bucket amzn-s3-demo-bucket \
--bucket-logging-status file:///logging.json
```

Isi dari `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"
      }
    ]
  }
}
```

`put-bucket-acl` perintah diperlukan untuk memberikan sistem pengiriman log S3 izin yang diperlukan (izin tulis dan baca-acp).

Untuk informasi selengkapnya, lihat [Logging Akses Server Amazon S3 di Panduan Pengembang Amazon S3](#).

- Untuk detail API, lihat [PutBucketLogging](#) di Referensi AWS CLI Perintah.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutBucketNotification** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan `PutBucketNotification`.

CLI

AWS CLI

Menerapkan konfigurasi notifikasi ke bucket bernama `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-notification --bucket amzn-s3-demo-bucket --notification-configuration file://notification.json
```

File `notification.json` ini adalah dokumen JSON di folder saat ini yang menentukan topik SNS dan jenis acara untuk dipantau:

```
{
  "TopicConfiguration": {
    "Event": "s3:ObjectCreated:*",
    "Topic": "arn:aws:sns:us-west-2:123456789012:s3-notification-topic"
  }
}
```

Topik SNS harus memiliki kebijakan IAM yang melekat padanya yang memungkinkan Amazon S3 untuk mempublikasikannya:

```
{
  "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"
        }
      }
    }
  ]
}
```

- Untuk detail API, lihat [PutBucketNotification](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Contoh ini mengonfigurasi konfigurasi topik SNS untuk acara S3 ObjectRemovedDelete dan mengaktifkan notifikasi untuk bucket s3 yang diberikan

```
$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
```

Contoh 2: Contoh ini memungkinkan pemberitahuan ObjectCreatedAll untuk bucket yang diberikan mengirimnya ke fungsi 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
```

Contoh 3: Contoh ini membuat 2 konfigurasi Lambda yang berbeda berdasarkan akhiran kunci yang berbeda dan dikonfigurasi keduanya dalam satu perintah.

```
#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:verifysm"
    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
```

- Untuk detail API, lihat [PutBucketNotification](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Contoh ini mengonfigurasi konfigurasi topik SNS untuk acara S3 ObjectRemovedDelete dan mengaktifkan notifikasi untuk bucket s3 yang diberikan

```
$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

```

Contoh 2: Contoh ini memungkinkan pemberitahuan ObjectCreatedAll untuk bucket yang diberikan mengirimnya ke fungsi 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

```

Contoh 3: Contoh ini membuat 2 konfigurasi Lambda yang berbeda berdasarkan akhiran kunci yang berbeda dan dikonfigurasi keduanya dalam satu perintah.

```

#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

```

- Untuk detail API, lihat [PutBucketNotification](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutBucketNotificationConfiguration** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan **PutBucketNotificationConfiguration**.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Memproses pemberitahuan acara S3](#)
- [Kirim pemberitahuan acara ke EventBridge](#)

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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}");
    }
}
}

```

- Untuk detail API, lihat [PutBucketNotificationConfiguration](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Untuk mengaktifkan notifikasi yang ditentukan ke bucket

put-bucket-notification-configuration Contoh berikut menerapkan konfigurasi notifikasi ke bucket bernama `amzn-s3-demo-bucket`. File tersebut `notification.json` adalah dokumen JSON di folder saat ini yang menentukan topik SNS dan jenis acara untuk dipantau.

```

aws s3api put-bucket-notification-configuration \
  --bucket amzn-s3-demo-bucket \
  --notification-configuration file://notification.json

```

Isi dari `notification.json`:

```

{
  "TopicConfigurations": [
    {
      "TopicArn": "arn:aws:sns:us-west-2:123456789012:s3-notification-
topic",
      "Events": [
        "s3:ObjectCreated:*"
      ]
    }
  ]
}

```

Topik SNS harus memiliki kebijakan IAM yang melekat padanya yang memungkinkan Amazon S3 untuk mempublikasikannya.

```
{
  "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"
        }
      }
    }
  ]
}
```

- Untuk detail API, lihat [PutBucketNotificationConfiguration](#) di Referensi AWS CLI Perintah.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutBucketPolicy** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan **PutBucketPolicy**.

C++

SDK untuk C++

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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"
        "}";
}

```

- Untuk detail API, lihat [PutBucketPolicy](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Contoh ini memungkinkan semua pengguna untuk mengambil objek apa pun di `amzn-s3-demo-bucket` kecuali yang ada di `MySecretFolder`. Ini juga memberikan `put` dan `delete` izin kepada pengguna `root` AWS akun `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/*"
    }
  ]
}

```

- Untuk detail API, lihat [PutBucketPolicy](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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();
}
}

```

- Untuk detail API, lihat [PutBucketPolicy](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Tambahkan kebijakan.

```
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;
  }
}
};
```

- Untuk informasi selengkapnya, lihat [AWS SDK untuk JavaScript Panduan Developer](#).
- Untuk detail API, lihat [PutBucketPolicy](#) di Referensi AWS SDK untuk JavaScript API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
```

- Untuk detail API, lihat [PutBucketPolicy](#) di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
```

- Untuk detail API, lihat [PutBucketPolicy](#) di Referensi AWS SDK untuk Ruby API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [PutBucketPolicy](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutBucketReplication** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan **PutBucketReplication**.

CLI

AWS CLI

Untuk mengonfigurasi replikasi untuk bucket S3

put-bucket-replicationContoh berikut menerapkan konfigurasi replikasi ke bucket S3 yang ditentukan.

```
aws s3api put-bucket-replication \  
  --bucket amzn-s3-demo-bucket1 \  
  --replication-configuration file://replication.json
```

Isi dari 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"  
      }  
    }  
  ]  
}
```

Bucket tujuan harus mengaktifkan versi. Peran yang ditentukan harus memiliki izin untuk menulis ke bucket tujuan dan memiliki hubungan kepercayaan yang memungkinkan Amazon S3 untuk mengambil peran tersebut.

Contoh kebijakan izin peran:

```
{  
  "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/*"
    }
  ]
}
```

Contoh kebijakan hubungan kepercayaan:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Perintah ini tidak menghasilkan output.

Untuk informasi selengkapnya, lihat [Ini adalah judul topik](#) di Panduan Pengguna Konsol Layanan Penyimpanan Sederhana Amazon.

- Untuk detail API, lihat [PutBucketReplication](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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());
}
}
```

- Untuk detail API, lihat [PutBucketReplication](#) di Referensi AWS SDK for Java 2.x API.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Contoh ini menetapkan konfigurasi replikasi dengan satu aturan yang memungkinkan replikasi ke bucket 'amzn-s3-demo-bucket' setiap objek baru yang dibuat dengan awalan nama kunci "" di 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
```

Contoh 2: Contoh ini menetapkan konfigurasi replikasi dengan beberapa aturan yang memungkinkan replikasi ke bucket 'amzn-s3-demo-bucket' setiap objek baru yang dibuat dengan awalan nama kunci "" atau "". TaxDocs OtherDocs Awalan kunci tidak boleh tumpang tindih.

```
$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

```

Contoh 3: Contoh ini memperbarui konfigurasi replikasi pada bucket yang ditentukan untuk menonaktifkan aturan yang mengontrol replikasi objek dengan awalan nama kunci "TaxDocs" ke 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

```

- Untuk detail API, lihat [PutBucketReplication](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Contoh ini menetapkan konfigurasi replikasi dengan satu aturan yang memungkinkan replikasi ke bucket 'amzn-s3-demo-bucket' setiap objek baru yang dibuat dengan awalan nama kunci "" di 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

```

Contoh 2: Contoh ini menetapkan konfigurasi replikasi dengan beberapa aturan yang memungkinkan replikasi ke bucket 'amzn-s3-demo-bucket' setiap objek baru yang dibuat dengan awalan nama kunci "" atau "". TaxDocs OtherDocs Awalan kunci tidak boleh tumpang tindih.

```

$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

```

Contoh 3: Contoh ini memperbarui konfigurasi replikasi pada bucket yang ditentukan untuk menonaktifkan aturan yang mengontrol replikasi objek dengan awalan nama kunci "TaxDocs" ke 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
```

- Untuk detail API, lihat [PutBucketReplication](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutBucketRequestPayment** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan **PutBucketRequestPayment**.

CLI

AWS CLI

Contoh 1: Untuk mengaktifkan konfigurasi ``requester pays`` untuk sebuah bucket

`put-bucket-request-payment` Contoh berikut memungkinkan requester pays untuk bucket yang ditentukan.

```
aws s3api put-bucket-request-payment \
```

```
--bucket amzn-s3-demo-bucket \  
--request-payment-configuration '{"Payer":"Requester"}'
```

Perintah ini tidak menghasilkan output.

Contoh 2: Untuk menonaktifkan konfigurasi ``requester pays`` untuk bucket

put-bucket-request-paymentContoh berikut menonaktifkan requester pays untuk bucket yang ditentukan.

```
aws s3api put-bucket-request-payment \  
--bucket amzn-s3-demo-bucket \  
--request-payment-configuration '{"Payer":"BucketOwner"}'
```

Perintah ini tidak menghasilkan output.

- Untuk detail API, lihat [PutBucketRequestPayment](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Memperbarui konfigurasi pembayaran permintaan untuk bucket bernama 'amzn-s3-demo-bucket' sehingga orang yang meminta unduhan dari bucket akan dikenakan biaya untuk unduhan. Secara default, pemilik bucket membayar unduhan. Untuk mengatur permintaan pembayaran kembali ke default gunakan 'BucketOwner' untuk parameter RequestPaymentConfiguration_Payer.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- Untuk detail API, lihat [PutBucketRequestPayment](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Memperbarui konfigurasi pembayaran permintaan untuk bucket bernama 'amzn-s3-demo-bucket' sehingga orang yang meminta unduhan dari bucket akan dikenakan biaya untuk unduhan. Secara default, pemilik bucket membayar unduhan. Untuk mengatur permintaan pembayaran kembali ke default gunakan 'BucketOwner' untuk parameter RequestPaymentConfiguration_Payer.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- Untuk detail API, lihat [PutBucketRequestPayment](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutBucketTagging** dengan CLI

Contoh kode berikut menunjukkan cara menggunakan **PutBucketTagging**.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Memulai dengan penyimpanan objek](#)

CLI

AWS CLI

Perintah berikut menerapkan konfigurasi penandaan ke bucket bernama `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging file://  
tagging.json
```

File tersebut `tagging.json` adalah dokumen JSON di folder saat ini yang menentukan tag:

```
{  
  "TagSet": [  
    {  
      "Key": "organization",  
      "Value": "marketing"  
    }  
  ]  
}
```

Atau terapkan konfigurasi penandaan `amzn-s3-demo-bucket` langsung dari baris perintah:

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging  
'TagSet=[{Key=organization,Value=marketing}]'
```

- Untuk detail API, lihat [PutBucketTagging](#) di Referensi AWS CLI Perintah.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini menerapkan dua tag ke bucket bernama **cloudtrail-test-2018**: tag dengan kunci Stage dan nilai Test, dan tag dengan kunci Environment dan nilai Alpha. Untuk memverifikasi bahwa tag telah ditambahkan ke bucket, jalankan **Get-S3BucketTagging -BucketName bucket_name**. Hasilnya harus menunjukkan tag yang Anda terapkan ke bucket di perintah pertama. Perhatikan bahwa **Write-S3BucketTagging** menimpa seluruh set tag yang ada di bucket. Untuk menambah atau menghapus tag individual, jalankan cmdlet Resource Groups dan Tagging API, dan **Add-RGResourceTag Remove-RGResourceTag**. Atau, gunakan Editor Tag di AWS Management Console untuk mengelola tag bucket S3.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";  
Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Contoh 2: Perintah ini menyalurkan ember yang diberi nama **cloudtrail-test-2018** ke dalam **Write-S3BucketTagging** cmdlet. Ini berlaku tag Stage:Production dan Department:Finance ke ember. Perhatikan bahwa **Write-S3BucketTagging** menimpa seluruh set tag yang ada di bucket.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging  
-TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";  
Value="Finance" } )
```

- Untuk detail API, lihat [PutBucketTagging](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini menerapkan dua tag ke bucket bernama **cloudtrail-test-2018**: tag dengan kunci Stage dan nilai Test, dan tag dengan kunci Environment dan nilai Alpha. Untuk

memverifikasi bahwa tag telah ditambahkan ke bucket, jalankan **Get-S3BucketTagging -BucketName bucket_name**. Hasilnya harus menunjukkan tag yang Anda terapkan ke bucket di perintah pertama. Perhatikan bahwa **Write-S3BucketTagging** menimpa seluruh set tag yang ada di bucket. Untuk menambah atau menghapus tag individual, jalankan cmdlet Resource Groups dan Tagging API, dan. **Add-RGTResourceTag Remove-RGTResourceTag** Atau, gunakan Editor Tag di AWS Management Console untuk mengelola tag bucket S3.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage"; Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Contoh 2: Perintah ini menyalurkan ember yang diberi nama **cloudtrail-test-2018** ke dalam **Write-S3BucketTagging** cmdlet. Ini berlaku tag Stage:Production dan Department:Finance ke ember. Perhatikan bahwa **Write-S3BucketTagging** menimpa seluruh set tag yang ada di bucket.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging -TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department"; Value="Finance" } )
```

- Untuk detail API, lihat [PutBucketTagging](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutBucketVersioning** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan **PutBucketVersioning**.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Memulai dengan penyimpanan objek](#)

CLI

AWS CLI

Perintah berikut memungkinkan pembuatan versi pada bucket bernama: `amzn-s3-demo-bucket`

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-configuration Status=Enabled
```

Perintah berikut memungkinkan pembuatan versi, dan menggunakan kode mfa

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-configuration Status=Enabled --mfa "SERIAL 123456"
```

- Untuk detail API, lihat [PutBucketVersioning](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);  
}
```

- Untuk detail API, lihat [PutBucketVersioning](#) di Referensi AWS SDK for Java 2.x API.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah memungkinkan pembuatan versi untuk bucket S3 yang diberikan.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -  
VersioningConfig_Status Enabled
```

- Untuk detail API, lihat [PutBucketVersioning](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah memungkinkan pembuatan versi untuk bucket S3 yang diberikan.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -  
VersioningConfig_Status Enabled
```

- Untuk detail API, lihat [PutBucketVersioning](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [PutBucketVersioning](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutBucketWebsite** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan **PutBucketWebsite**.

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

```
// Put the website configuration.
PutBucketWebsiteRequest putRequest = new
PutBucketWebsiteRequest()
{
    BucketName = bucketName,
    WebsiteConfiguration = new WebsiteConfiguration()
    {
        IndexDocumentSuffix = indexDocumentSuffix,
        ErrorDocument = errorDocument,
    },
};
PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);
```

- Untuk detail API, lihat [PutBucketWebsite](#) di Referensi AWS SDK untuk .NET API.

C++

SDK untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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();
}

```

- Untuk detail API, lihat [PutBucketWebsite](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Menerapkan konfigurasi situs web statis ke bucket bernama `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-website --bucket amzn-s3-demo-bucket --website-configuration file://website.json
```

File tersebut `website.json` adalah dokumen JSON di folder saat ini yang menentukan halaman indeks dan kesalahan untuk situs web:

```
{
  "IndexDocument": {
    "Suffix": "index.html"
  },
  "ErrorDocument": {
    "Key": "error.html"
  }
}
```

- Untuk detail API, lihat [PutBucketWebsite](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
}
}
```

- Untuk detail API, lihat [PutBucketWebsite](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Atur konfigurasi situs 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;  
  }  
}  
};
```

- Untuk informasi selengkapnya, lihat [AWS SDK untuk JavaScript Panduan Developer](#).
- Untuk detail API, lihat [PutBucketWebsite](#) di Referensi AWS SDK untuk JavaScript API.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah memungkinkan hosting situs web untuk bucket yang diberikan dengan dokumen indeks sebagai 'index.html' dan dokumen kesalahan sebagai 'error.html'.

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'  
-WebsiteConfiguration_IndexDocumentSuffix 'index.html' -  
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Untuk detail API, lihat [PutBucketWebsite](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah memungkinkan hosting situs web untuk bucket yang diberikan dengan dokumen indeks sebagai 'index.html' dan dokumen kesalahan sebagai 'error.html'.

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'  
-WebsiteConfiguration_IndexDocumentSuffix 'index.html' -  
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Untuk detail API, lihat [PutBucketWebsite](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Ruby

SDK untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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__
```

- Untuk detail API, lihat [PutBucketWebsite](#) di Referensi AWS SDK untuk Ruby API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutObject** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `PutObject`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)
- [Memulai dengan penyimpanan objek](#)
- [Buat permintaan bersyarat](#)
- [Lacak unggahan dan unduhan](#)
- [Bekerja dengan integritas objek Amazon S3](#)

.NET

SDK untuk .NET (v4)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
    }
}
```

- Untuk detail API, lihat [PutObject](#) di Referensi AWS SDK untuk .NET API.

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Unggah objek dengan enkripsi di sisi 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");
    }
}
```

```

    }
  }
}

```

Letakkan objek menggunakan permintaan bersyarat.

```

/// <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;
    }
}

```

- Untuk detail API, lihat [PutObject](#) di Referensi AWS SDK untuk .NET API.

Bash

AWS CLI dengan skrip Bash

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat [PutObject](#) di Referensi AWS CLI Perintah.

C++

SDK untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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();
}

```

- Untuk detail API, lihat [PutObject](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Contoh 1: Unggah objek ke Amazon S3

Contoh put-object perintah berikut mengunggah objek ke Amazon S3.

```
aws s3api put-object \
```

```
--bucket amzn-s3-demo-bucket \  
--key my-dir/MySampleImage.png \  
--body MySampleImage.png
```

Untuk informasi selengkapnya tentang mengunggah objek, lihat Mengunggah Objek < <http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html> > di Panduan Pengembang Amazon S3.

Contoh 2: Unggah file video ke Amazon S3

Contoh put-object perintah berikut mengunggah 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
```

Untuk informasi selengkapnya tentang mengunggah objek, lihat Mengunggah Objek < <http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html> > di Panduan Pengembang Amazon S3.

- Untuk detail API, lihat [PutObject](#) di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Masukkan objek ke dalam ember dengan menggunakan API tingkat rendah.

```
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
}

```

Unggah objek ke bucket dengan menggunakan manajer transfer.

```

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
}
```

- Untuk detail API, lihat [PutObject](#) di Referensi AWS SDK untuk Go API.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Unggah file ke bucket menggunakan [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);
        }
    });
}
```

Gunakan file [S3TransferManager](#) untuk [mengunggah file](#) ke ember. Lihat [file lengkap](#) dan [lakukan pengujian](#).

```
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();
    }
```

Unggah objek ke bucket dan tetapkan tanda menggunakan [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;
}
```

```
}  
}
```

Unggah objek ke bucket dan tetapkan metadata menggunakan [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);
    }
}
}

```

Unggah objek ke bucket dan tetapkan nilai retensi objek menggunakan [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);
        }
    }
}

```

- Untuk detail API, lihat [PutObject](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Unggah objek tersebut.

```

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;
    }
  }
};

```

Unggah objek dengan syarat ETag cocok dengan yang disediakan.

```

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"));
  }
}
```

- Untuk informasi selengkapnya, silakan lihat [Panduan Developer AWS SDK untuk JavaScript](#).
- Untuk detail API, lihat [PutObject](#) di Referensi AWS SDK untuk JavaScript API.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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}")
    }
}
```

- Untuk detail API, lihat [PutObject](#) di AWS SDK untuk referensi API Kotlin.

PHP

SDK untuk PHP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Unggah objek ke 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.");
}
```

- Untuk detail API, lihat [PutObject](#) di Referensi AWS SDK untuk PHP API.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah ini mengunggah file tunggal "local-sample.txt" ke Amazon S3, membuat objek dengan kunci "sample.txt" di bucket "test-files".

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

Contoh 2: Perintah ini mengunggah file tunggal "sample.txt" ke Amazon S3, membuat objek dengan kunci "sample.txt" di bucket "test-files". Jika parameter -Key tidak disediakan, nama file digunakan sebagai kunci objek S3.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Contoh 3: Perintah ini mengunggah file tunggal "local-sample.txt" ke Amazon S3, membuat objek dengan kunci prefix/to "/sample.txt" di bucket "test-files".

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -  
File .\local-sample.txt
```

Contoh 4: Perintah ini mengunggah semua file di subdirektori "Scripts" ke bucket "test-files" dan menerapkan common key prefix "" untuk setiap objek. SampleScripts Setiap file yang diunggah akan memiliki kunci "SampleScripts/filename" di mana 'nama file' bervariasi.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\
```

Contoh 5: Perintah ini mengunggah semua file*.ps1 di direktur lokal "Scripts" ke bucket "test-files" dan menerapkan common key prefix "" ke setiap objek. SampleScripts Setiap file yang diunggah akan memiliki kunci "SampleScripts/filename.ps1" di mana 'nama file' bervariasi.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\ -SearchPattern *.ps1
```

Contoh 6: Perintah ini membuat objek S3 baru yang berisi string konten tertentu dengan kunci 'sample.txt'.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object  
contents"
```

Contoh 7: Perintah ini mengunggah file yang ditentukan (nama file digunakan sebagai kunci) dan menerapkan tag yang ditentukan ke objek baru.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet  
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Contoh 8: Perintah ini secara rekursif mengunggah folder yang ditentukan dan menerapkan tag yang ditentukan ke semua objek baru.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"
-Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Untuk detail API, lihat [PutObject](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah ini mengunggah file tunggal "local-sample.txt" ke Amazon S3, membuat objek dengan kunci "sample.txt" di bucket "test-files".

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

Contoh 2: Perintah ini mengunggah file tunggal "sample.txt" ke Amazon S3, membuat objek dengan kunci "sample.txt" di bucket "test-files". Jika parameter -Key tidak disediakan, nama file digunakan sebagai kunci objek S3.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Contoh 3: Perintah ini mengunggah file tunggal "local-sample.txt" ke Amazon S3, membuat objek dengan kunci prefix/to "/sample.txt" di bucket "test-files".

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -
File .\local-sample.txt
```

Contoh 4: Perintah ini mengunggah semua file di subdirektori "Scripts" ke bucket "test-files" dan menerapkan common key prefix "" untuk setiap objek. SampleScripts Setiap file yang diunggah akan memiliki kunci "SampleScripts/filename" di mana 'nama filen' bervariasi.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
SampleScripts\
```

Contoh 5: Perintah ini mengunggah semua file*.ps1 di direktur lokal "Scripts" ke bucket "test-files" dan menerapkan common key prefix "" ke setiap objek. SampleScripts Setiap file yang diunggah akan memiliki kunci "SampleScripts/filename.ps1" di mana 'nama filen' bervariasi.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\ -SearchPattern *.ps1
```

Contoh 6: Perintah ini membuat objek S3 baru yang berisi string konten tertentu dengan kunci 'sample.txt'.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object  
contents"
```

Contoh 7: Perintah ini mengunggah file yang ditentukan (nama file digunakan sebagai kunci) dan menerapkan tag yang ditentukan ke objek baru.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet  
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Contoh 8: Perintah ini secara rekursif mengunggah folder yang ditentukan dan menerapkan tag yang ditentukan ke semua objek baru.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"  
-Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Untuk detail API, lihat [PutObject](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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()
```

Unggah objek menggunakan permintaan bersyarat.

```
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
```

- Untuk detail API, lihat [PutObject](#) di AWS SDK for Python (Boto3) Referensi API.

Ruby

SDK untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Unggah file menggunakan uploader terkelola (`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__

```

Unggah file menggunakan 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__

```

Unggah file menggunakan `Object.put` dan menambahkan enkripsi sisi 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__
```

- Untuk detail API, lihat [PutObject](#) di Referensi AWS SDK untuk Ruby API.

Rust

SDK for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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)  
}
```

- Untuk detail API, lihat [PutObject](#) referensi AWS SDK for Rust API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [PutObject](#) di AWS SDK untuk referensi SAP ABAP API.

Swift

SDK para Swift

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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  
}  
}
```

- Untuk detail API, lihat referensi [PutObject AWSSDK](#) untuk Swift API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutObjectAcl** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan **PutObjectAcl**.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Mengelola daftar kontrol akses \(ACL\)](#)

C++

SDK untuk C++

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
}

```

- Untuk detail API, lihat [PutObjectAcl](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Perintah berikut memberikan full control kepada dua AWS pengguna (user1@example.com dan user2@example.com) dan read izin untuk semua orang:

```

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

```

Lihat <http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html> detail tentang ACL kustom (perintah s3api ACL, seperti put-object-acl, menggunakan notasi argumen singkatan yang sama).

- Untuk detail API, lihat [PutObjectAcl](#) di Referensi AWS CLI Perintah.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

```

- Untuk detail API, lihat [PutObjectAcl](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [PutObjectAcl](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutObjectLegalHold** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan **PutObjectLegalHold**.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Kunci objek Amazon S3](#)

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
    }
}
```

```
}
```

- Untuk detail API, lihat [PutObjectLegalHold](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Untuk menerapkan Penahanan Hukum pada suatu objek

`put-object-legal-hold` Contoh berikut menetapkan Penahanan Hukum pada objek `doc1.rtf`.

```
aws s3api put-object-legal-hold \  
  --bucket amzn-s3-demo-bucket-with-object-lock \  
  --key doc1.rtf \  
  --legal-hold Status=ON
```

Perintah ini tidak menghasilkan output.

- Untuk detail API, lihat [PutObjectLegalHold](#) di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat [PutObjectLegalHold](#) di Referensi AWS SDK untuk Go API.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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 +".");
}
```

- Untuk detail API, lihat [PutObjectLegalHold](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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"));
    }
}
}

```

- Untuk detail API, lihat [PutObjectLegalHold](#) di Referensi AWS SDK untuk JavaScript API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Letakkan benda penahanan hukum.

```

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
    )

```

- Untuk detail API, lihat [PutObjectLegalHold](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [PutObjectLegalHold](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutObjectLockConfiguration** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `PutObjectLockConfiguration`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Kunci objek Amazon S3](#)

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Atur konfigurasi kunci objek dari ember.

```
/// <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;
    }
}

```

Setel periode retensi default 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($"{bucketName}");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"{ex.Message}");
        return false;
    }
}
```

- Untuk detail API, lihat [PutObjectLockConfiguration](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Untuk mengatur konfigurasi kunci objek pada ember

put-object-lock-configuration Contoh berikut menetapkan kunci objek 50 hari pada bucket yang ditentukan.

```
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 } } }'
```

Perintah ini tidak menghasilkan output.

- Untuk detail API, lihat [PutObjectLockConfiguration](#) di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Atur konfigurasi kunci objek dari ember.

```
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
}

```

Setel periode retensi default 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
}
```

- Untuk detail API, lihat [PutObjectLockConfiguration](#) di Referensi AWS SDK untuk Go API.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Atur konfigurasi kunci objek dari ember.

```
// 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()
+ "'");
    }
}
```

```
}  
}
```

Setel periode retensi default 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
+ ".");
}
```

- Untuk detail API, lihat [PutObjectLockConfiguration](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Atur konfigurasi kunci objek dari ember.

```
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"));
  }
}

```

Setel periode retensi default 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"));  
  }  
}
```

- Untuk detail API, lihat [PutObjectLockConfiguration](#) di Referensi AWS SDK untuk JavaScript API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Letakkan konfigurasi kunci objek.

```
s3_client.put_object_lock_configuration(
    Bucket=bucket,
    ObjectLockConfiguration={"ObjectLockEnabled": "Disabled", "Rule":
    {}},
)
```

- Untuk detail API, lihat [PutObjectLockConfiguration](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [PutObjectLockConfiguration](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutObjectRetention** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan **PutObjectRetention**.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Kunci objek Amazon S3](#)

.NET

SDK untuk .NET

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
}
}

```

- Untuk detail API, lihat [PutObjectRetention](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Untuk mengatur konfigurasi retensi objek untuk objek

put-object-retention Contoh berikut menetapkan konfigurasi retensi objek untuk objek yang ditentukan hingga 2025-01-01.

```

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" }'

```

Perintah ini tidak menghasilkan output.

- Untuk detail API, lihat [PutObjectRetention](#) di Referensi AWS CLI Perintah.

Go

SDK untuk Go V2

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
}

```

- Untuk detail API, lihat [PutObjectRetention](#) di Referensi AWS SDK untuk Go API.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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 +".");
}
```

- Untuk detail API, lihat [PutObjectRetention](#) di Referensi AWS SDK for Java 2.x API.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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"));
  }
}
```

- Untuk detail API, lihat [PutObjectRetention](#) di Referensi AWS SDK untuk JavaScript API.

PowerShell

Alat untuk PowerShell V4

Contoh 1: Perintah mengaktifkan mode retensi tata kelola hingga tanggal '31 Des 2019 00:00:00' untuk objek 'testfile.txt' di bucket S3 yang diberikan.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -  
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Untuk detail API, lihat [PutObjectRetention](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V4).

Alat untuk PowerShell V5

Contoh 1: Perintah mengaktifkan mode retensi tata kelola hingga tanggal '31 Des 2019 00:00:00' untuk objek 'testfile.txt' di bucket S3 yang diberikan.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -  
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Untuk detail API, lihat [PutObjectRetention](#) di Referensi Alat AWS untuk PowerShell Cmdlet (V5).

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Letakkan retensi objek.

```
s3_client.put_object_retention(  
    Bucket=bucket,  
    Key=key,  
    VersionId=version_id,
```

```

        Retention={"Mode": "GOVERNANCE", "RetainUntilDate":
far_future_date},
        BypassGovernanceRetention=True,
    )

```

- Untuk detail API, lihat [PutObjectRetention](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.

```

- Untuk detail API, lihat [PutObjectRetention](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **RestoreObject** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `RestoreObject`.

.NET

SDK untuk .NET

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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}");
}
}
```

- Untuk detail API, lihat [RestoreObject](#) di Referensi AWS SDK untuk .NET API.

CLI

AWS CLI

Untuk membuat permintaan pemulihan untuk objek

`restore-object` Contoh berikut mengembalikan objek Amazon S3 Glacier yang ditentukan untuk bucket selama 10 hari. `my-glacier-bucket`

```
aws s3api restore-object \
  --bucket my-glacier-bucket \
  --key doc1.rtf \
  --restore-request Days=10
```

Perintah ini tidak menghasilkan output.

- Untuk detail API, lihat [RestoreObject](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
    }
}

```

- Untuk detail API, lihat [RestoreObject](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **SelectObjectContent** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `SelectObjectContent`.

CLI

AWS CLI

Untuk memfilter konten objek Amazon S3 berdasarkan pernyataan SQL

`select-object-content` Contoh berikut menyaring objek `my-data-file.csv` dengan pernyataan SQL tertentu dan mengirimkan output ke 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"
```

Perintah ini tidak menghasilkan output.

- Untuk detail API, lihat [SelectObjectContent](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Contoh berikut menunjukkan query menggunakan objek JSON. [Contoh lengkap](#) juga menunjukkan penggunaan objek 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:
{} \n bytesProcessed: {} \n bytesReturned: {}",
                    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;
    }
}
```

- Untuk detail API, lihat [SelectObjectContent](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **UploadPart** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `UploadPart`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Gunakan checksum](#)
- [Bekerja dengan integritas objek Amazon S3](#)

C++

SDK untuk C++

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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,

```

```

std::shared_ptr<Aws::IOStream> &body,
const
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);
}

```

- Untuk detail API, lihat [UploadPart](#) di Referensi AWS SDK untuk C++ API.

CLI

AWS CLI

Perintah berikut mengunggah bagian pertama dalam unggahan multibagian yang dimulai dengan perintah: `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"
```

bodyOpsi ini mengambil nama atau jalur file lokal untuk diunggah (jangan gunakan awalan `file://`). Ukuran bagian minimum adalah 5 MB. Upload ID dikembalikan oleh `create-multipart-upload` dan juga dapat diambil dengan `list-multipart-uploads`. Bucket dan kunci ditentukan saat Anda membuat unggahan multipart.

Output:

```
{
  "ETag": "\"e868e0f4719e394144ef36531ee6824c\""
}
```

Simpan nilai ETag dari setiap bagian untuk nanti. Mereka diminta untuk menyelesaikan unggahan multipart.

- Untuk detail API, lihat [UploadPart](#) di Referensi AWS CLI Perintah.

Rust

SDK for Rust

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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?;
```

- Untuk detail API, lihat [UploadPart](#) referensi AWS SDK for Rust API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **UploadPartCopy** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `UploadPartCopy`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Melakukan penyalinan multibagian](#)

CLI

AWS CLI

Untuk meng-upload bagian dari objek dengan menyalin data dari objek yang ada sebagai sumber data

upload-part-copyContoh berikut mengunggah bagian dengan menyalin data dari objek yang ada sebagai sumber data.

```
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\""  
  }  
}
```

- Untuk detail API, lihat [UploadPartCopy](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
}

```

- Untuk detail API, lihat [UploadPartCopy](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Skenario untuk Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menerapkan skenario umum di Amazon S3 dengan AWS SDK. Skenario ini menunjukkan kepada Anda cara menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam Amazon S3 atau digabungkan dengan yang lain. Layanan AWS Setiap skenario menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode.

Skenario menargetkan tingkat pengalaman menengah untuk membantu Anda memahami tindakan layanan dalam konteks.

Contoh

- [Periksa apakah ada ember](#)
- [Periksa apakah ada objek](#)
- [Mengkonversi teks menjadi ucapan dan kembali ke teks menggunakan AWS SDK](#)
- [Buat URL yang telah ditetapkan sebelumnya untuk Amazon S3 menggunakan AWS SDK](#)
- [Membuat aplikasi manajemen aset foto yang memungkinkan pengguna mengelola foto menggunakan label](#)
- [Halaman web yang mencantumkan objek Amazon S3 menggunakan AWS SDK](#)
- [Membuat aplikasi penjelajah Amazon Textract](#)
- [Hapus semua objek dalam bucket Amazon S3 tertentu menggunakan AWS SDK](#)
- [Hapus unggahan multipart yang tidak lengkap ke Amazon S3 menggunakan AWS SDK](#)
- [Mendeteksi APD dalam gambar dengan Amazon Rekognition menggunakan AWS SDK](#)
- [Mendeteksi entitas dalam teks yang diekstraksi dari gambar menggunakan AWS SDK](#)
- [Mendeteksi wajah dalam gambar menggunakan AWS SDK](#)
- [Mendeteksi objek dalam gambar dengan Amazon Rekognition menggunakan AWS SDK](#)
- [Mendeteksi orang dan objek dalam video dengan Amazon Rekognition menggunakan AWS SDK](#)
- [Unduh 'direktori' S3 dari bucket Amazon Simple Storage Service \(Amazon S3\)](#)
- [Mengunduh semua objek di bucket Amazon Simple Storage Service \(Amazon S3\) ke direktori lokal](#)
- [Unduh aliran dengan ukuran yang tidak diketahui dari objek Amazon S3 menggunakan AWS SDK](#)

- [Dapatkan objek Amazon S3 dari Multi-Region Access Point dengan menggunakan AWS SDK](#)
- [Dapatkan objek dari bucket Amazon S3 menggunakan AWS SDK, menentukan header If-Modified-Since](#)
- [Memulai enkripsi untuk objek Amazon S3 menggunakan AWS SDK](#)
- [Memulai tag untuk objek Amazon S3 menggunakan AWS SDK](#)
- [Memulai dengan cluster pemrosesan data besar](#)
- [Memulai dengan manajemen konfigurasi](#)
- [Memulai dengan ekstraksi teks dokumen](#)
- [Memulai dengan toko fitur pembelajaran mesin](#)
- [Memulai dengan penyimpanan objek](#)
- [Memulai dengan analisis kueri](#)
- [Bekerja dengan fitur kunci objek Amazon S3 menggunakan AWS SDK](#)
- [Buat permintaan bersyarat Amazon S3 menggunakan AWS SDK](#)
- [Mengelola daftar kontrol akses \(ACL\) untuk bucket Amazon S3 menggunakan AWS SDK](#)
- [Mengelola pesan Amazon SQS besar menggunakan Amazon S3 dengan AWS SDK](#)
- [Mengelola objek Amazon S3 berversi dalam batch dengan fungsi Lambda menggunakan AWS SDK](#)
- [Mengurai URI Amazon S3 menggunakan AWS SDK](#)
- [Melakukan salinan multipart dari objek Amazon S3 menggunakan AWS SDK](#)
- [Menerima dan memproses notifikasi acara Amazon S3 dengan menggunakan AWS SDK](#)
- [Simpan EXIF dan informasi gambar lainnya menggunakan AWS SDK](#)
- [Kirim pemberitahuan acara S3 ke Amazon EventBridge menggunakan AWS SDK](#)
- [Lacak unggahan atau unduh objek Amazon S3 menggunakan AWS SDK](#)
- [Transformasi data untuk aplikasi Anda dengan S3 Object Lambda](#)
- [Contoh pendekatan untuk pengujian unit dan integrasi dengan AWS SDK](#)
- [Mengunggah direktori lokal secara rekursif ke bucket Amazon Simple Storage Service \(Amazon S3\)](#)
- [Unggah atau unduh file besar ke dan dari Amazon S3 menggunakan AWS SDK](#)
- [Unggah aliran dengan ukuran yang tidak diketahui ke objek Amazon S3 menggunakan AWS SDK](#)
- [Gunakan checksum untuk bekerja dengan objek Amazon S3 menggunakan AWS SDK](#)
- [Bekerja dengan fitur integritas objek Amazon S3 menggunakan AWS SDK](#)

- [Bekerja dengan objek berversi Amazon S3 menggunakan AWS SDK](#)

Periksa apakah ada ember

Contoh kode berikut menunjukkan cara memeriksa apakah ada bucket.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Anda dapat menggunakan `doesBucketExists` metode berikut sebagai pengganti metode 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() == HttpStatus.MOVED_PERMANENTLY) ||
"AccessDenied".equals(ase.awsErrorDetails().errorCode())) {
            return true;
        }
        if (ase.statusCode() == HttpStatus.NOT_FOUND) {
            return false;
        }
    }
}

```

```
        throw ase;
    }
}
}
```

- Untuk detail API, lihat [GetBucketAcl](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Periksa apakah ada objek

Contoh kode berikut menunjukkan bagaimana untuk memeriksa apakah objek ada.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Anda dapat menggunakan `doesObjectExist` metode berikut sebagai pengganti metode 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;  
    }  
}  
}
```

- Untuk detail API, lihat [HeadObject](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mengkonversi teks menjadi ucapan dan kembali ke teks menggunakan AWS SDK

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Gunakan Amazon Polly untuk mensintesis file input teks biasa (UTF-8) ke file audio.
- Mengunggah file audio ke bucket Amazon S3.
- Menggunakan Amazon Transcribe untuk mengonversi file audio menjadi teks.
- Tampilkan teks.

Rust

SDK for Rust

Gunakan Amazon Polly untuk mensintesis file input teks biasa (UTF-8) ke file audio, mengunggah file audio ke bucket Amazon S3, menggunakan Amazon Transcribe untuk mengonversi file audio tersebut menjadi teks, dan menampilkan teks.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Polly
- Amazon S3
- Amazon Transcribe

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Buat URL yang telah ditetapkan sebelumnya untuk Amazon S3 menggunakan AWS SDK

Contoh kode berikut ini menunjukkan cara membuat URL yang telah ditandatangani untuk Amazon S3 dan mengunggah objek.

.NET

SDK untuk .NET

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat URL yang telah ditetapkan sebelumnya yang dapat melakukan tindakan Amazon S3 untuk waktu yang terbatas.

```
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;
    }
}
```

Buat URL yang telah ditetapkan sebelumnya dan unggah menggunakan URL tersebut.

```
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 untuk .NET (v4)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Membuat dan menggunakan URL POST presigned untuk upload browser langsung.

```
/// <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 untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hasilkan URL yang telah ditandatangani sebelumnya untuk mengunduh objek.

```
//! 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);
}

```

Unduh menggunakan 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;
}

```

Buat URL yang telah ditandatangani sebelumnya untuk mengunggah objek.

```

//! 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);
}

```

Unggah menggunakan 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 untuk Go V2

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat fungsi yang membungkus tindakan S3 yang telah ditetapkan sebelumnya.

```
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
}

```

Jalankan contoh interaktif yang menghasilkan dan menggunakan URL yang telah ditetapkan sebelumnya untuk mengunggah, mengunduh, dan menghapus objek 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/awsddocs/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))
}
```

Tentukan pembungkus permintaan HTTP yang digunakan oleh contoh untuk membuat permintaan 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 untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Berikut ini menunjukkan tiga contoh cara membuat URL presigned dan menggunakan URL dengan perpustakaan klien HTTP:

- Permintaan HTTP GET yang menggunakan URL dengan tiga pustaka klien HTTP
- Permintaan HTTP PUT dengan metadata di header yang menggunakan URL dengan tiga pustaka klien HTTP
- Permintaan HTTP PUT dengan parameter kueri yang menggunakan URL dengan satu pustaka klien HTTP

Buat URL yang telah ditandatangani sebelumnya untuk suatu objek, lalu unduh (GET request).

Impor.

```
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;
```

Hasilkan 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();
    }
}

```

Unduh objek dengan menggunakan salah satu dari tiga pendekatan berikut.

Gunakan kelas JDK `URLConnection` (sejak v1.1) untuk melakukan download.

```

/* 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();
}
```

Gunakan kelas JDK `HttpClient` (sejak v11) untuk melakukan download.

```
/* 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();
}
```

Gunakan kelas AWS SDK for Java `SdkHttpClient` Java untuk melakukan 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();
    }
}

```

Buat URL yang telah ditandatangani sebelumnya dengan metadata di header untuk diunggah, lalu unggah file (permintaan PUT).

Impor.

```

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;
```

Hasilkan 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();
    }
}

```

Unggah objek file dengan menggunakan salah satu dari tiga pendekatan berikut.

Gunakan kelas JDK `URLConnection` (sejak v1.1) untuk melakukan upload.

```

/* 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);
}
}

```

Gunakan kelas JDK `HttpClient` (sejak v11) untuk melakukan upload.

```

/* 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);  
    }  
}
```

Gunakan `SdkHttpClient` kelas AWS for Java V2 untuk melakukan upload.

```
/* 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);  
    }  
}
```

Buat URL yang telah ditandatangani sebelumnya dengan parameter kueri untuk unggahan, lalu unggah file (permintaan PUT).

Impor.

```
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;
```

Hasilkan 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();
    }
}
```

Gunakan `SdkHttpClient` kelas AWS for Java V2 untuk melakukan upload.

```
/**
 * 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 untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat URL yang telah ditetapkan sebelumnya untuk unggah objek ke 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;
    }
  }
};
```

Buat URL yang telah ditetapkan sebelumnya untuk mengunduh objek dari 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;
  }
}
};
```

- Untuk informasi selengkapnya, lihat [AWS SDK untuk JavaScript Panduan Developer](#).

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat permintaan `GetObject` yang telah ditetapkan sebelumnya dan gunakan URL untuk mengunduh objek.

```
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
}
```

Buat permintaan yang `GetObject` telah ditetapkan sebelumnya dengan opsi lanjutan.

```
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
}
```

Buat permintaan `PutObject` yang telah ditetapkan sebelumnya dan gunakan untuk mengunggah objek.

```
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)
}
```

- Untuk informasi selengkapnya, lihat [AWS panduan pengembang SDK untuk Kotlin](#).

PHP

SDK untuk PHP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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 untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat URL yang telah ditetapkan sebelumnya yang dapat melakukan tindakan S3 untuk waktu yang terbatas. Gunakan paket Permintaan untuk membuat permintaan dengan 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()

```

Buat permintaan POST yang telah ditetapkan sebelumnya untuk mengunggah 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):
        """
        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
AWS
        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 untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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 for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat permintaan presigning ke objek GET S3.

```
/// 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(())
}
```

Buat permintaan presigning ke objek PUT S3.

```
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 for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat permintaan yang telah ditetapkan sebelumnya ke objek GET S3.

```

" 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( ).

```

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Membuat aplikasi manajemen aset foto yang memungkinkan pengguna mengelola foto menggunakan label

Contoh kode berikut ini menunjukkan cara membuat aplikasi nirserver yang memungkinkan pengguna mengelola foto menggunakan label.

.NET

SDK untuk .NET

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK untuk C++

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK untuk Java 2.x

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK untuk JavaScript (v3)

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

- API Gateway
- DynamoDB
- Lambda

- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK untuk Kotlin

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK untuk PHP

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK untuk Rust

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Untuk mendalami tentang asal usul contoh ini, lihat postingan di [Komunitas AWS](#).

Layanan yang digunakan dalam contoh ini

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Halaman web yang mencantumkan objek Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara mendaftar objek Amazon S3 di halaman web.

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Kode berikut adalah komponen React yang relevan yang membuat panggilan ke AWS SDK. Versi runnable dari aplikasi yang berisi komponen ini dapat ditemukan di link sebelumnya 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;
```

- Untuk detail API, lihat [ListObjects](#) di Referensi AWS SDK untuk JavaScript API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Membuat aplikasi penjelajah Amazon Textract

Contoh kode berikut ini menunjukkan cara menjelajahi output Amazon Textract melalui aplikasi interaktif.

JavaScript

SDK untuk JavaScript (v3)

Menunjukkan cara menggunakan aplikasi AWS SDK untuk JavaScript untuk membangun aplikasi React yang menggunakan Amazon Textract untuk mengekstrak data dari gambar dokumen dan menampilkannya di halaman web interaktif. Contoh ini berjalan di peramban web dan memerlukan identitas Amazon Cognito yang diautentikasi sebagai kredensialnya. Contoh ini menggunakan Amazon Simple Storage Service (Amazon S3) untuk penyimpanan, dan untuk notifikasi, contoh ini mengambil polling antrean Amazon Simple Queue Service (Amazon SQS) yang berlangganan topik Amazon Simple Notification Service (Amazon SNS).

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Identitas Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK untuk Python (Boto3)

Menunjukkan cara menggunakan Amazon Textract untuk mendeteksi elemen teks, formulir, dan tabel dalam gambar dokumen. AWS SDK untuk Python (Boto3) Gambar input dan output Amazon Textract ditampilkan dalam aplikasi Tkinter yang memungkinkan Anda menjelajahi elemen yang terdeteksi.

- Kirim gambar dokumen ke Amazon Textract dan jelajahi output elemen yang terdeteksi.
- Kirim gambar langsung ke Amazon Textract atau melalui bucket Amazon Simple Storage Service (Amazon S3).
- Gunakan API asinkron untuk memulai tugas yang menerbitkan notifikasi ke topik Amazon Simple Notification Service (Amazon SNS) saat tugas selesai.
- Lakukan polling pada antrean Amazon Simple Queue Service (Amazon SQS) untuk mendapatkan pesan penyelesaian tugas dan tampilkan hasilnya.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Identitas Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Hapus semua objek dalam bucket Amazon S3 tertentu menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menghapus semua objek di bucket Amazon S3.

JavaScript

SDK untuk JavaScript (v3)

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hapus semua objek untuk bucket Amazon S3 tertentu.

```
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);
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk JavaScript .
 - [DeleteObjects](#)
 - [ListObjectsV2](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Hapus unggahan multipart yang tidak lengkap ke Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menghapus atau menghentikan unggahan multipart Amazon S3 yang tidak lengkap.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Untuk menghentikan unggahan multibagian yang sedang berlangsung atau tidak lengkap karena alasan apa pun, Anda bisa mendapatkan unggahan daftar lalu menghapusnya seperti yang ditunjukkan pada contoh berikut.

```
/**
 * 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);
            }
        }
    }
}

```

Untuk menghapus unggahan multibagian yang tidak lengkap yang dimulai sebelum atau sesudah tanggal, Anda dapat menghapus unggahan multibagian secara selektif berdasarkan titik waktu seperti yang ditunjukkan pada contoh berikut.

```

/**
 * 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);
            }
        }
    }
}

```

Jika Anda memiliki akses ke ID unggahan setelah memulai unggahan multibagian, Anda dapat menghapus unggahan yang sedang berlangsung dengan menggunakan 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);
        }
    }
}

```

Untuk secara konsisten menghapus unggahan multibagian yang tidak lengkap yang lebih lama dalam beberapa hari tertentu, siapkan konfigurasi siklus hidup bucket untuk bucket. Contoh berikut menunjukkan cara membuat aturan untuk menghapus unggahan yang tidak lengkap yang lebih lama dari 7 hari.

```

/**
 * 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());
        }
    }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .
 - [AbortMultipartUpload](#)
 - [ListMultipartUploads](#)
 - [PutBucketLifecycleConfiguration](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mendeteksi APD dalam gambar dengan Amazon Rekognition menggunakan AWS SDK

Contoh kode berikut menunjukkan cara membuat aplikasi yang menggunakan Amazon Rekognition untuk mendeteksi Alat Pelindung Diri (APD) dalam gambar.

Java

SDK untuk Java 2.x

Menunjukkan cara membuat AWS Lambda fungsi yang mendeteksi gambar dengan Alat Pelindung Diri.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- DynamoDB
- Amazon Rekognition

- Amazon S3
- Amazon SES

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mendeteksi entitas dalam teks yang diekstraksi dari gambar menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan Amazon Comprehend untuk mendeteksi entitas dalam teks yang diekstrak oleh Amazon Textract dari gambar yang disimpan di Amazon S3.

Python

SDK untuk Python (Boto3)

Menunjukkan cara menggunakan AWS SDK untuk Python (Boto3) dalam buku catatan Jupyter untuk mendeteksi entitas dalam teks yang diekstraksi dari gambar. Contoh ini menggunakan Amazon Textract untuk mengekstrak teks dari gambar yang disimpan di Amazon Simple Storage Service (Amazon S3) dan Amazon Comprehend untuk mendeteksi entitas dalam teks yang diekstraksi.

Contoh ini adalah notebook Jupyter dan harus dijalankan di lingkungan yang dapat meng-host notebook. Untuk petunjuk tentang cara menjalankan contoh menggunakan Amazon SageMaker AI, lihat petunjuk di [TextractAndComprehendNotebook.ipynb](#).

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Comprehend
- Amazon S3
- Amazon Textract

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mendeteksi wajah dalam gambar menggunakan AWS SDK

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Menyimpan gambar di bucket Amazon S3.
- Menggunakan Amazon Rekognition untuk mendeteksi detail wajah, seperti rentang usia, jenis kelamin, dan emosi (seperti tersenyum).
- Menampilkan detail tersebut.

Rust

SDK untuk Rust

Menyimpan gambar di bucket Amazon S3 dengan prefiks unggahan, menggunakan Amazon Rekognition untuk mendeteksi detail wajah, seperti rentang usia, jenis kelamin, dan emosi (tersenyum, dll.), dan menampilkan detail tersebut.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon S3

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mendeteksi objek dalam gambar dengan Amazon Rekognition menggunakan AWS SDK

Contoh kode berikut menunjukkan cara membuat aplikasi yang menggunakan Amazon Rekognition untuk mendeteksi objek berdasarkan kategori dalam gambar.

.NET

SDK untuk .NET

Menunjukkan cara menggunakan Amazon Rekognition .NET API untuk membuat aplikasi yang menggunakan Amazon Rekognition untuk mengidentifikasi objek berdasarkan kategori dalam gambar yang berada di bucket Amazon Simple Storage Service (Amazon S3). Aplikasi ini mengirimkan notifikasi email kepada admin beserta hasilnya menggunakan Amazon Simple Email Service (Amazon SES).

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK untuk Java 2.x

Menunjukkan cara menggunakan Amazon Rekognition Java API untuk membuat aplikasi yang menggunakan Amazon Rekognition untuk mengidentifikasi objek berdasarkan kategori dalam gambar yang terletak di bucket Amazon Simple Storage Service (Amazon S3). Aplikasi ini mengirimkan notifikasi email kepada admin beserta hasilnya menggunakan Amazon Simple Email Service (Amazon SES).

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK untuk JavaScript (v3)

Menunjukkan cara menggunakan Amazon Rekognition dengan membuat aplikasi AWS SDK untuk JavaScript yang menggunakan Amazon Rekognition untuk mengidentifikasi objek berdasarkan kategori dalam gambar yang terletak di bucket Amazon Simple Storage Service (Amazon S3). Aplikasi ini mengirimkan notifikasi email kepada admin beserta hasilnya menggunakan Amazon Simple Email Service (Amazon SES).

Pelajari cara:

- Membuat pengguna yang tidak diautentikasi menggunakan Amazon Cognito.
- Menganalisis gambar untuk objek menggunakan Amazon Rekognition.
- Memverifikasi alamat email untuk Amazon SES.
- Mengirim notifikasi email menggunakan Amazon SES.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK untuk Kotlin

Menunjukkan cara menggunakan Amazon Rekognition Kotlin API untuk membuat aplikasi yang menggunakan Amazon Rekognition untuk mengidentifikasi objek berdasarkan kategori dalam gambar yang berada di bucket Amazon Simple Storage Service (Amazon S3). Aplikasi ini mengirimkan notifikasi email kepada admin beserta hasilnya menggunakan Amazon Simple Email Service (Amazon SES).

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK untuk Python (Boto3)

Menunjukkan cara menggunakan AWS SDK untuk Python (Boto3) untuk membuat aplikasi web yang memungkinkan Anda melakukan hal berikut:

- Mengunggah foto ke bucket Amazon Simple Storage Service (Amazon S3).
- Menggunakan Amazon Rekognition untuk menganalisis dan memberi label pada foto.
- Menggunakan Amazon Simple Email Service (Amazon SES) untuk mengirim laporan email analisis gambar.

Contoh ini berisi dua komponen utama: halaman web yang ditulis di dalamnya JavaScript yang dibangun dengan React, dan layanan REST yang ditulis dengan Python yang dibangun dengan Flask-RESTful

Anda dapat menggunakan halaman web React untuk:

- Menampilkan daftar gambar yang disimpan di bucket S3 Anda.
- Mengunggah gambar dari komputer ke bucket S3.
- Menampilkan gambar dan label yang mengidentifikasi item yang terdeteksi dalam gambar.
- Mendapatkan laporan semua gambar di bucket S3 Anda dan mengirimkan email laporan tersebut.

Halaman web memanggil layanan REST. Layanan mengirimkan permintaan ke AWS untuk melakukan tindakan berikut:

- Mendapatkan dan memfilter daftar gambar dalam bucket S3 Anda.
- Mengunggah foto ke bucket S3 Anda.
- Menggunakan Amazon Rekognition untuk menganalisis foto individual dan mendapatkan daftar label yang mengidentifikasi item yang terdeteksi dalam foto.
- Menganalisis semua foto di bucket S3 Anda dan menggunakan Amazon SES untuk mengirim laporan melalui email.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon S3
- Amazon SES

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mendeteksi orang dan objek dalam video dengan Amazon Rekognition menggunakan AWS SDK

Contoh kode berikut menunjukkan cara mendeteksi orang dan objek dalam video dengan Amazon Rekognition.

Java

SDK untuk Java 2.x

Menunjukkan cara menggunakan Amazon Rekognition Java API untuk membuat aplikasi guna mendeteksi wajah dan objek di video yang berada di bucket Amazon Simple Storage Service (Amazon S3). Aplikasi ini mengirimkan notifikasi email kepada admin beserta hasilnya menggunakan Amazon Simple Email Service (Amazon SES).

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK untuk Python (Boto3)

Gunakan Amazon Rekognition untuk mendeteksi wajah, objek, dan orang dalam video dengan memulai tugas deteksi asinkron. Contoh ini juga mengonfigurasi Amazon Rekognition untuk memberi tahu topik Amazon Simple Notification Service (Amazon SNS) saat pekerjaan selesai dan berlangganan antrian Amazon Simple Queue Service (Amazon SQS) ke topik tersebut. Ketika antrian menerima pesan tentang pekerjaan, pekerjaan diambil dan hasilnya adalah output.

Contoh ini paling baik dilihat di GitHub. Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Unduh 'direktori' S3 dari bucket Amazon Simple Storage Service (Amazon S3)

Contoh kode berikut menunjukkan cara mengunduh dan memfilter konten 'direktori' bucket Amazon S3.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Contoh ini menunjukkan cara menggunakan [S3TransferManager](#)in AWS SDK for Java 2.x untuk mengunduh 'direktori' dari bucket Amazon S3. Ini juga menunjukkan bagaimana menggunakan [DownloadFilters](#)dalam permintaan.

```
/**
 * 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;
    }

```

```
}
```

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mengunduh semua objek di bucket Amazon Simple Storage Service (Amazon S3) ke direktori lokal

Contoh Kode berikut ini menunjukkan cara mengunduh semua objek di bucket Amazon Simple Storage Service (Amazon S3) ke direktori lokal.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Gunakan file [S3TransferManager](#) untuk [mengunduh semua objek S3](#) di bucket S3 yang sama. Lihat [file lengkap](#) dan [lakukan pengujian](#).

```
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();
}
```

- Untuk detail API, lihat [DownloadDirectory](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Unduh aliran dengan ukuran yang tidak diketahui dari objek Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara mengunduh aliran dengan ukuran yang tidak diketahui dari objek Amazon S3.

Swift

SDK para Swift

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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 {
        throw TransferError.downloadError("Error: No data returned for
download")
    }

    // 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).")
}

```

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Dapatkan objek Amazon S3 dari Multi-Region Access Point dengan menggunakan AWS SDK

Contoh kode berikut menunjukkan bagaimana untuk mendapatkan objek dari Multi-Region Access Point.

Kotlin

SDK untuk Kotlin

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Konfigurasi klien S3 untuk menggunakan algoritma penandatanganan 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  
}
```

Gunakan Multi-Region Access Point ARN sebagai ganti nama bucket untuk mengambil objek.

```
suspend fun getObjectFromMrp(  
    s3: S3Client,  
    mrpArn: String,  
    keyName: String,  
): String? {  
    val request = GetObjectRequest {  
        bucket = mrpArn // 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 $mrarn")
        }
    }
    return stringObj
}
```

- Untuk informasi selengkapnya, lihat [AWS panduan pengembang SDK untuk Kotlin](#).
- Untuk detail API, lihat [GetObject](#) di AWS SDK untuk referensi API Kotlin.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Dapatkan objek dari bucket Amazon S3 menggunakan AWS SDK, menentukan header If-Modified-Since

Contoh kode berikut menunjukkan cara membaca data dari objek di bucket S3, tetapi hanya jika bucket tersebut belum dimodifikasi sejak waktu pengambilan terakhir.

Rust

SDK for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh selengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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(())
}

```

- Untuk detail API, lihat [GetObject](#) referensi AWS SDK for Rust API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Memulai enkripsi untuk objek Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara memulai enkripsi untuk objek Amazon S3.

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di [Repositori Contoh Kode 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>

```

```

S3    /// <param name="keyName">The name of the object to upload to the Amazon
    /// 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);
    }
}

```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk .NET .

- [CopyObject](#)
- [GetObject](#)
- [GetObjectMetadata](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Memulai tag untuk objek Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara memulai tag untuk objek Amazon S3.

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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}'");
            }
        }
    }
}
```

- Untuk detail API, lihat [GetObjectTagging](#) di Referensi AWS SDK untuk .NET API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Memulai dengan cluster pemrosesan data besar

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat key pair EC2
- Siapkan penyimpanan dan siapkan aplikasi Anda
- Pembersihan sumber daya

Bash

AWS CLI dengan skrip Bash

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankan di repositori [tutorial pengembang Sample](#).

```
#!/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..."
    fi
}
```

```
        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

    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
```



```
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."
```

- Untuk detail API, lihat topik berikut di Referensi Perintah AWS CLI .
 - [AddSteps](#)
 - [Cp](#)
 - [CreateCluster](#)
 - [CreateDefaultRoles](#)
 - [CreateKeyPair](#)
 - [DescribeCluster](#)
 - [DescribeKeyPairs](#)
 - [DescribeStep](#)
 - [Ls](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)
 - [Ssh](#)
 - [TerminateClusters](#)
 - [Tunggu](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Memulai dengan manajemen konfigurasi

Contoh kode berikut ini menunjukkan cara untuk melakukan:

Skenario

- Buat bucket Amazon S3.
- Membuat topik Amazon SNS
- Buat peran IAM untuk Config
- Siapkan perekam konfigurasi Config
- Siapkan saluran pengiriman Config
- Mulai perekam konfigurasi
- Verifikasi pengaturan Config

Bash

AWS CLI dengan skrip Bash

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankan di repositori [tutorial pengembang Sample](#).

```
#!/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
    fi
}
```

```
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

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="$AWS_REGION")
    fi
    check_command "$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!"
```

- Untuk detail API, lihat topik berikut di Referensi Perintah AWS CLI .

- [AttachRolePolicy](#)
- [CreateBucket](#)
- [CreateRole](#)
- [CreateTopic](#)
- [DeleteBucket](#)
- [DeleteConfigurationRecorder](#)
- [DeleteDeliveryChannel](#)
- [DeleteRole](#)
- [DeleteRolePolicy](#)
- [DeleteTopic](#)
- [DescribeConfigurationRecorderStatus](#)
- [DescribeConfigurationRecorders](#)
- [DescribeDeliveryChannels](#)
- [DetachRolePolicy](#)
- [GetCallerIdentity](#)
- [PutConfigurationRecorder](#)
- [PutDeliveryChannel](#)
- [PutPublicAccessBlock](#)
- [PutRolePolicy](#)

- [Rm](#)
- [StartConfigurationRecorder](#)
- [StopConfigurationRecorder](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Memulai dengan ekstraksi teks dokumen

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat Bucket S3
- Unggah dokumen ke S3
- Pembersihan sumber daya

Bash

AWS CLI dengan skrip Bash

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankan di repositori [tutorial pengembang Sample](#).

```
#!/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..."
    fi
}
```



```
        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 $AWS_REGION..."
TEXTTRACT_CHECK=$(aws textract help 2>&1)
TEXTTRACT_CHECK_STATUS=$?
if [ $TEXTTRACT_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: $DOCUMENT_NAME"
else
    # Fallback: create a minimal valid PNG using base64
    echo "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAyAAAAfFcSJAADU1EQVR42mNk
+M9QDwADhgGAWjR9awAAAABJRU5ErkJggg==" | base64 -d > "$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
```

- Untuk detail API, lihat topik berikut di Referensi Perintah AWS CLI .
 - [AnalyzeDocument](#)
 - [Cp](#)
 - [Bantuan](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Memulai dengan toko fitur pembelajaran mesin

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Siapkan izin IAM

- Buat peran SageMaker eksekusi
- Buat grup fitur
- Pembersihan sumber daya

Bash

AWS CLI dengan skrip Bash

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankan di repositori [tutorial pengembang Sample](#).

```
#!/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"
                ;;
        esac
    done
}
```



```

        "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
```

```

    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)"

```

- Untuk detail API, lihat topik berikut di Referensi Perintah AWS CLI .

- [AttachRolePolicy](#)
- [CreateBucket](#)
- [CreateFeatureGroup](#)
- [CreateRole](#)
- [DeleteBucket](#)
- [DeleteFeatureGroup](#)
- [DeleteRole](#)

- [DescribeFeatureGroup](#)
- [DetachRolePolicy](#)
- [GetCallerIdentity](#)
- [GetRole](#)
- [ListFeatureGroups](#)
- [PutPublicAccessBlock](#)
- [Rm](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Memulai dengan penyimpanan objek

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat bucket S3 pertama Anda
- Mengunggah objek
- Aktifkan versioning
- Konfigurasi enkripsi default
- Tambahkan tag ke ember Anda
- Daftar objek dan versi
- Pembersihan sumber daya

Bash

AWS CLI dengan skrip Bash

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankan di repositori [tutorial pengembang Sample](#).

```
#!/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
    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 default encryption"
    exit 1
fi
echo "SSE-S3 default encryption configured."
```

```

    }
  ]
}' >/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."

```

- Untuk detail API, lihat topik berikut di [Referensi Perintah AWS CLI](#) .

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [HeadObject](#)
- [ListObjectVersions](#)
- [ListObjectsV2](#)
- [PutBucketEncryption](#)
- [PutBucketTagging](#)
- [PutBucketVersioning](#)
- [PutObject](#)
- [PutPublicAccessBlock](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Memulai dengan analisis kueri

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat bucket S3 untuk hasil kueri
- Buat database
- Membuat tabel
- Jalankankueri
- Membuat dan menggunakan kueri bernama
- Pembersihan sumber daya

Bash

AWS CLI dengan skrip Bash

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankan di repositori [tutorial pengembang Sample](#).

```
#!/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
}
```

```

    fi
    if [ -n "${DATABASE_NAME:-}" ]; then
        echo "- Database: $DATABASE_NAME"
        if [ -n "${TABLE_NAME:-}" ]; then
            echo "- Table: $TABLE_NAME in $DATABASE_NAME"
        fi
    fi
    fi
    if [ -n "${S3_BUCKET:-}" ]; then
        echo "- S3 Bucket: $S3_BUCKET"
    fi
    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."
    fi
done
```

```
        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 (
```

[illegible]

```
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
    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

    # 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://$S3_BUCKET" 2>&1)
        if echo "$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
    fi

```

```
        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!"
```

- Untuk detail API, lihat topik berikut di Referensi Perintah AWS CLI .
 - [Cp](#)
 - [CreateNamedQuery](#)
 - [DeleteNamedQuery](#)
 - [GetNamedQuery](#)
 - [GetQueryExecution](#)
 - [GetQueryResults](#)
 - [ListDatabases](#)
 - [ListNamedQueries](#)
 - [ListTableMetadata](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)
 - [StartQueryExecution](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Bekerja dengan fitur kunci objek Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara bekerja dengan fitur kunci objek S3.

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Jalankan skenario interaktif yang menunjukkan fitur kunci objek 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;
}
}
```

Kelas pembungkus untuk fungsi 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($"\\tAdded an object lock policy to bucket
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"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($"\\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($"\\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();
    }
}

/// <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(
        $"Deleted {objectKey} in {bucketName}.");
    return true;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"\\tUnable 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($"\\tDelete for {bucketName} complete.");
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to delete bucket {bucketName}: " +
ex.Message);
        return false;
    }
}
```

```
}  
  
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk .NET .
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

Go

SDK untuk Go V2

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Jalankan skenario interaktif yang menunjukkan fitur kunci objek 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\n(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))
}
```

Tentukan struct yang membungkus tindakan S3 yang digunakan dalam contoh ini.

```
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
}

```

Pembersihan sumber daya

```

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)
}
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk Go .
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Jalankan skenario interaktif yang menunjukkan fitur kunci objek 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;
}
}
```

Kelas pembungkus untuk fungsi 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.waitForBucketExists(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.getVersionId());
                        s3InfoObject.setKeyName(version.getKey());
                        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());
}
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Entrypoint untuk skenario ()index.js. Ini mengatur semua langkah. Kunjungi GitHub untuk melihat detail implementasi untuk Skenario ScenarioInput, ScenarioOutput,, dan 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]",
  });
}

```

Keluarkan pesan selamat datang ke konsol (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 };
```

Menyebarkan bucket, objek, dan pengaturan 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,
};
```

Melihat dan menghapus file di 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 };

```

Hancurkan semua sumber daya yang dibuat (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 };
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk JavaScript .
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)

- [PutObjectLegalHold](#)
- [PutObjectLockConfiguration](#)
- [PutObjectRetention](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Buat permintaan bersyarat Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menambahkan prasyarat ke permintaan Amazon S3.

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Jalankan skenario interaktif yang mendemonstrasikan fitur permintaan bersyarat 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(new ObjectKey,
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;
    }
}

```

Kelas pembungkus untuk fungsi 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;
    }
}
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk .NET .
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

JavaScript

SDK untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Entrypoint untuk alur kerja (). `index.js` Ini mengatur semua langkah. Kunjungi GitHub untuk melihat detail implementasi untuk Skenario `ScenarioInput`, `ScenarioOutput`, dan `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]",
  });
}
```

Keluarkan pesan selamat datang ke konsol (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 };

```

Menyebarkan ember dan objek ()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,
};
```

Dapatkan, salin, dan letakkan objek menggunakan permintaan bersyarat 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 };

```

Hancurkan semua sumber daya yang dibuat (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 };
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk JavaScript .
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Jalankan skenario interaktif yang menunjukkan permintaan bersyarat 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,
condition_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()
```

Kelas pembungkus yang mendefinisikan operasi permintaan bersyarat.

```
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
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk Python (Boto3).
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mengelola daftar kontrol akses (ACL) untuk bucket Amazon S3 menggunakan AWS SDK

Contoh kode berikut ini menunjukkan cara mengelola daftar kontrol akses (ACL) untuk bucket Amazon S3.

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di [Repositori Contoh Kode 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>

```

```

S3    /// <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
    /// 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    /// 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    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,
});
}

}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk .NET .
 - [GetBucketAcl](#)
 - [GetObjectAcl](#)
 - [PutBucketAcl](#)
 - [PutObjectAcl](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mengelola pesan Amazon SQS besar menggunakan Amazon S3 dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan Amazon SQS Extended Client Library untuk bekerja dengan pesan Amazon SQS besar.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
}
```

- Untuk informasi selengkapnya, silakan lihat [Panduan Developer AWS SDK for Java 2.x](#).
- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .
 - [CreateBucket](#)
 - [PutBucketLifecycleConfiguration](#)
 - [ReceiveMessage](#)
 - [SendMessage](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mengelola objek Amazon S3 berversi dalam batch dengan fungsi Lambda menggunakan AWS SDK

Contoh kode berikut ini menunjukkan cara mengelola objek S3 berversi dalam batch dengan fungsi Lambda.

Python

SDK untuk Python (Boto3)

Menunjukkan cara memanipulasi objek berversi Amazon Simple Storage Service (Amazon S3) dalam batch dengan membuat pekerjaan yang memanggil fungsi untuk melakukan pemrosesan. AWS Lambda Contoh ini membuat bucket yang diaktifkan versinya, mengunggah bait dari puisi You Are Old, Father William oleh Lewis Carroll, dan menggunakan pekerjaan batch Amazon S3 untuk mengubah puisi dengan berbagai cara.

Pelajari cara:

- Membuat fungsi Lambda yang beroperasi pada objek berversi.
- Membuat manifes objek untuk diperbarui.
- Membuat pekerjaan batch yang menginvokasi fungsi Lambda untuk memperbarui objek.
- Menghapus fungsi Lambda.
- Mengosongkan dan menghapus bucket berversi.

Contoh ini paling baik dilihat di GitHub. Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Amazon S3

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mengurai URI Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara mengurai URI Amazon S3 untuk mengekstrak komponen penting seperti nama bucket dan kunci objek.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Mengurai Amazon S3 URI dengan menggunakan kelas [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);
        }
    }
}

```

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Melakukan salinan multipart dari objek Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara melakukan penyalinan multibagian dari sebuah objek Amazon S3.

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di [Repositori Contoh Kode 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");
}
}
}

```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk .NET .

- [CompleteMultipartUpload](#)
- [CreateMultipartUpload](#)
- [GetObjectMetadata](#)
- [UploadPartCopy](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Menerima dan memproses notifikasi acara Amazon S3 dengan menggunakan AWS SDK

Contoh kode berikut menunjukkan cara bekerja dengan pemberitahuan acara S3 dengan cara berorientasi objek.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Contoh ini menunjukkan cara memproses acara notifikasi S3 dengan menggunakan 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);
}
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .
 - [DeleteMessageBatch](#)
 - [GetQueueAttributes](#)
 - [PutBucketNotificationConfiguration](#)
 - [ReceiveMessage](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Simpan EXIF dan informasi gambar lainnya menggunakan AWS SDK

Contoh kode berikut ini menunjukkan cara:

- Mendapatkan informasi EXIF dari file JPG, JPEG, atau PNG.
- Mengunggah file gambar ke bucket Amazon S3.
- Menggunakan Amazon Rekognition untuk mengidentifikasi tiga atribut teratas (label) dalam file.

- Menambahkan informasi EXIF dan label ke tabel Amazon DynamoDB di Wilayah.

Rust

SDK untuk Rust

Mendapatkan informasi EXIF dari file JPG, JPEG, atau PNG, mengunggah file gambar ke bucket Amazon S3, menggunakan Amazon Rekognition untuk mengidentifikasi tiga atribut terbatas (label di Amazon Rekognition) dalam file, dan menambahkan EXIF dan informasi label ke tabel Amazon DynamoDB di Wilayah.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- DynamoDB
- Amazon Rekognition
- Amazon S3

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Kirim pemberitahuan acara S3 ke Amazon EventBridge menggunakan AWS SDK

Contoh kode berikut menunjukkan cara mengaktifkan bucket untuk mengirim notifikasi peristiwa S3 ke EventBridge dan merutekan notifikasi ke topik Amazon SNS dan antrean Amazon SQS.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di [Repositori Contoh Kode 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"]
                            }
                        }
                    }
                """).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;
}

```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .
 - [PutBucketNotificationConfiguration](#)
 - [PutRule](#)
 - [PutTargets](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Lacak unggahan atau unduh objek Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara melacak unggahan atau unduhan objek Amazon S3.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Lacak kemajuan unggahan 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!
    */
}

```

Lacak kemajuan unduhan 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!

        */
    }

```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .
 - [GetObject](#)
 - [PutObject](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Transformasi data untuk aplikasi Anda dengan S3 Object Lambda

Contoh kode berikut menunjukkan cara mengubah data untuk aplikasi Anda dengan S3 Object Lambda.

.NET

SDK untuk .NET

Menunjukkan cara menambahkan kode kustom ke permintaan GET S3 standar untuk memodifikasi objek yang diminta diambil dari S3 sehingga objek sesuai dengan kebutuhan klien atau aplikasi yang meminta.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di [GitHub](#).

Layanan yang digunakan dalam contoh ini

- Lambda
- Amazon S3

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Contoh pendekatan untuk pengujian unit dan integrasi dengan AWS SDK

Contoh kode berikut menunjukkan cara contoh teknik praktik terbaik saat menulis unit dan pengujian integrasi menggunakan AWS SDK.

Rust

SDK for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Cargo.toml untuk contoh pengujian.

```
[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"

```

Contoh pengujian unit menggunakan automock dan pembungkus layanan.

```

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);
}
}

```

Contoh pengujian integrasi menggunakan 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",
        "atestsecretkey",
        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(&[]);
}
}
```

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Mengunggah direktori lokal secara rekursif ke bucket Amazon Simple Storage Service (Amazon S3)

Contoh kode berikut ini menunjukkan cara mengunggah direktori lokal secara rekursif ke bucket Amazon Simple Storage Service (Amazon S3).

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Gunakan [S3TransferManager](#) untuk [mengunggah direktori lokal](#). Lihat [file lengkap](#) dan [lakukan pengujian](#).

```
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();
    }
```

- Untuk detail API, lihat [UploadDirectory](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Unggah atau unduh file besar ke dan dari Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara mengunggah atau mengunduh file besar ke dan dari Amazon S3.

Untuk informasi selengkapnya, lihat [Pengunggahan objek menggunakan unggahan multibagian](#).

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Fungsi panggilan yang mentransfer file ke dan dari bucket S3 menggunakan Amazon TransferUtility S3.

```
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);
}

```

Unggah file tunggal.

```

    /// <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;
}
}
```

Unggah seluruh direktori lokal.

```
/// <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;
}
}

```

Unduh file tunggal.

```

/// <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}"));
}

```

Unduh konten 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;
    }
}
```

Lacak kemajuan unggahan menggunakan file 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}");
}
}
```

Unggah objek dengan enkripsi.

```
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);
    }

    /// <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 abortMPURquest = new
AbortMultipartUploadRequest
        {
```

```
        BucketName = existingBucketName,  
        Key = sourceKeyName,  
        UploadId = initResponse.UploadId,  
    };  
  
    await client.AbortMultipartUploadAsync(abortMPURequest);  
}  
}  
}
```

Go

SDK untuk Go V2

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat fungsi yang menggunakan pengelola unggah dan unduhan untuk memecah data menjadi beberapa bagian dan mentransfernya secara bersamaan.

```
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
}

```

Jalankan skenario interaktif yang menunjukkan cara menggunakan pengelola unggahan dan unduhan dalam konteks.

```

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 untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Fungsi panggilan yang mentransfer file ke dan dari bucket S3 menggunakan file.

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();
}
```

Unggah seluruh direktori lokal.

```
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();
}
```

Unggah file tunggal.

```
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();
}
```

Contoh kode menggunakan impor berikut.

```
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;
```

Gunakan [S3 Transfer Manager](#) di atas [klien AWS CRT-based S3](#) untuk melakukan unggahan multipart secara transparan ketika ukuran konten melebihi ambang batas. Ukuran ambang default adalah 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();
}
```

Gunakan [S3Client API](#) untuk melakukan upload multipart.

```
/**
 * 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()));

```

```
}
```

Gunakan [S3AsyncClient API](#) dengan dukungan multipart diaktifkan untuk melakukan pengunggahan multipart.

```
/**
 * 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 untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Unggah file besar.

```
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;
    }
}
};

```

Unduh file besar.

```

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 untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat fungsi yang mentransfer file menggunakan beberapa pengaturan manajer transfer yang tersedia. Gunakan kelas panggilan balik untuk menulis progres callback selama transfer 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

```

Menunjukkan fungsi manajer transfer dan melaporkan hasil.

```

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"Demonstration 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 for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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(())
}
```

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Unggah aliran dengan ukuran yang tidak diketahui ke objek Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara mengunggah aliran dengan ukuran yang tidak diketahui ke objek Amazon S3.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Gunakan [Klien 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;
    }
}
```

Gunakan [klien S3 asinkron standar dengan](#) unggahan multibagian diaktifkan.

```
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;
}
}
```

Gunakan [Manajer Transfer Amazon 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.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 para Swift

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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).")
}
```

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan checksum untuk bekerja dengan objek Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan checksum untuk bekerja dengan objek Amazon S3.

Java

SDK untuk Java 2.x

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Contoh kode menggunakan subset dari impor berikut.

```
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;
```

Tentukan algoritma checksum untuk metode `putObject` saat Anda [membangun `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"));
}
```

Verifikasi checksum untuk `getObject` metode saat Anda [membangun `GetObjectRequest`](#)

```
public GetObjectResponse getObjectWithChecksum(String bucketName, String key)
{
    return s3Client.getObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumMode(ChecksumMode.ENABLED))
        .response();
}
```

Pre-calculate checksum untuk `putObject` metode saat Anda [membangun `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)));
}
```

Gunakan [S3 Transfer Manager](#) di atas [klien AWS CRT-based S3](#) untuk melakukan unggahan multipart secara transparan ketika ukuran konten melebihi ambang batas. Ukuran ambang default adalah 8 MB.

Anda dapat menentukan algoritma checksum untuk SDK yang akan digunakan. Secara default, SDK menggunakan algoritma 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();
}
```

Gunakan [S3Client API](#) atau ([S3AsyncClient API](#)) untuk melakukan unggahan multipart. Jika Anda menentukan checksum tambahan, Anda juga harus menentukan algoritma yang akan digunakan pada inisiasi unggahan. Anda juga harus menentukan algoritma untuk setiap permintaan bagian dan memberikan checksum yang dihitung untuk setiap bagian setelah diunggah.


```

    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()));
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [UploadPart](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Bekerja dengan fitur integritas objek Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara bekerja dengan fitur integritas objek S3.

C++

SDK untuk C++

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Jalankan skenario interaktif yang menunjukkan fitur integritas objek 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;
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk C++ .
 - [AbortMultipartUpload](#)
 - [CompleteMultipartUpload](#)

- [CreateMultipartUpload](#)
- [DeleteObject](#)
- [GetObjectAttributes](#)
- [PutObject](#)
- [UploadPart](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Bekerja dengan objek berversi Amazon S3 menggunakan AWS SDK

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Membuat bucket S3 berversi.
- Mendapatkan semua versi objek.
- Mengembalikan objek ke versi sebelumnya.
- Menghapus dan memulihkan objek berversi.
- Menghapus semua versi objek secara permanen.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat fungsi yang membungkus tindakan 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

```

Unggah bait puisi ke objek berversi dan lakukan serangkaian tindakan pada objek tersebut.

```

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!")
```

- Untuk detail API, lihat topik berikut ini adalah Referensi API SDK untuk Python (Boto3)AWS
 - [CreateBucket](#)
 - [DeleteObject](#)
 - [ListObjectVersions](#)
 - [PutBucketLifecycleConfiguration](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Contoh tanpa server untuk Amazon S3

Contoh kode berikut menunjukkan cara menggunakan Amazon S3 dengan AWS SDK.

Contoh

- [Menginvokasi fungsi Lambda dari pemicu Amazon S3](#)

Menginvokasi fungsi Lambda dari pemicu Amazon S3

Contoh kode berikut menunjukkan cara mengimplementasikan fungsi Lambda yang menerima peristiwa yang dipicu dengan mengunggah objek ke bucket S3. Fungsi ini mengambil nama bucket S3 dan kunci objek dari parameter peristiwa dan memanggil Amazon S3 API untuk mengambil dan mencatat jenis konten objek.

.NET

SDK untuk .NET

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

Menggunakan peristiwa S3 dengan Lambda menggunakan .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 untuk Go V2

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

Menggunakan peristiwa S3 dengan Lambda menggunakan 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 untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

Menggunakan peristiwa S3 dengan Lambda menggunakan 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 untuk JavaScript (v3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

Mengkonsumsi acara S3 dengan menggunakan JavaScript Lambda.

```
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);
    }
};
```

Mengkonsumsi acara S3 dengan menggunakan TypeScript Lambda.

```
// 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 untuk PHP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

Mengkonsumsi acara S3 dengan Lambda menggunakan 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 untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

Menggunakan peristiwa S3 dengan Lambda menggunakan 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 untuk Ruby

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

Mengkonsumsi acara S3 dengan Lambda menggunakan 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 for Rust

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori [contoh Nirserver](#).

Menggunakan peristiwa S3 dengan Lambda menggunakan 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(())
}
```

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Contoh kode untuk Amazon S3 Control menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan Kontrol Amazon S3 dengan kit pengembangan AWS perangkat lunak (SDK).

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Kontrol Amazon S3

- [Contoh dasar untuk Amazon S3 Control menggunakan AWS SDK](#)
 - [Halo Kontrol Amazon S3](#)
 - [Pelajari dasar-dasar Kontrol Amazon S3 dengan AWS SDK](#)
 - [Tindakan untuk Kontrol Amazon S3 menggunakan AWS SDK](#)
 - [Gunakan CreateJobdengan AWS SDK atau CLI](#)
 - [Gunakan DeleteJobTaggingdengan AWS SDK](#)
 - [Gunakan DescribeJobdengan AWS SDK atau CLI](#)
 - [Gunakan GetJobTaggingdengan AWS SDK](#)
 - [Gunakan PutJobTaggingdengan AWS SDK](#)
 - [Gunakan UpdateJobPrioritydengan AWS SDK atau CLI](#)
 - [Gunakan UpdateJobStatusdengan AWS SDK atau CLI](#)

Contoh dasar untuk Amazon S3 Control menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan dasar-dasar Kontrol Amazon S3 dengan AWS SDK.

Contoh

- [Halo Kontrol Amazon S3](#)
- [Pelajari dasar-dasar Kontrol Amazon S3 dengan AWS SDK](#)
- [Tindakan untuk Kontrol Amazon S3 menggunakan AWS SDK](#)
 - [Gunakan CreateJobdengan AWS SDK atau CLI](#)
 - [Gunakan DeleteJobTaggingdengan AWS SDK](#)
 - [Gunakan DescribeJobdengan AWS SDK atau CLI](#)
 - [Gunakan GetJobTaggingdengan AWS SDK](#)

- [Gunakan PutJobTaggingdengan AWS SDK](#)
- [Gunakan UpdateJobPrioritydengan AWS SDK atau CLI](#)
- [Gunakan UpdateJobStatusdengan AWS SDK atau CLI](#)

Halo Kontrol Amazon S3

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon S3 Control.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
    });
}

```

- Untuk detail API, lihat [ListJobs](#) di Referensi AWS SDK for Java 2.x API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
```

- Untuk detail API, lihat [ListJobs](#) di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Pelajari dasar-dasar Kontrol Amazon S3 dengan AWS SDK

Contoh kode berikut menunjukkan cara mempelajari operasi inti untuk Kontrol Amazon S3.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Pelajari operasi inti.

```
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.");
        }
    }
}
}
}

```

Kelas aksi yang membungkus operasi.

```
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();
}
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .

- [CreateJob](#)
- [DeleteJobTagging](#)
- [DescribeJob](#)
- [GetJobTagging](#)
- [ListJobs](#)
- [PutJobTagging](#)
- [UpdateJobPriority](#)
- [UpdateJobStatus](#)

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Pelajari Skenario Dasar Batch S3.

```
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
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk Python (Boto3).
 - [CreateJob](#)
 - [DeleteJobTagging](#)
 - [DescribeJob](#)
 - [GetJobTagging](#)
 - [ListJobs](#)
 - [PutJobTagging](#)
 - [UpdateJobPriority](#)
 - [UpdateJobStatus](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Tindakan untuk Kontrol Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara melakukan tindakan Kontrol Amazon S3 individual dengan AWS SDK. Setiap contoh menyertakan tautan ke GitHub, di mana Anda dapat menemukan instruksi untuk mengatur dan menjalankan kode.

Contoh berikut hanya mencakup tindakan yang paling umum digunakan. Untuk daftar lengkapnya, lihat Referensi [API Kontrol Amazon S3](#).

Contoh

- [Gunakan CreateJobdengan AWS SDK atau CLI](#)
- [Gunakan DeleteJobTaggingdengan AWS SDK](#)
- [Gunakan DescribeJobdengan AWS SDK atau CLI](#)
- [Gunakan GetJobTaggingdengan AWS SDK](#)
- [Gunakan PutJobTaggingdengan AWS SDK](#)
- [Gunakan UpdateJobPrioritydengan AWS SDK atau CLI](#)
- [Gunakan UpdateJobStatusdengan AWS SDK atau CLI](#)

Gunakan **CreateJob**dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakanCreateJob.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)

CLI

AWS CLI

Untuk membuat pekerjaan operasi batch Amazon S3

create-jobContoh berikut membuat pekerjaan operasi batch Amazon S3 untuk menandai objek sebagai `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"
}
```

- Untuk detail API, lihat [CreateJob](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat pekerjaan S3 asinkron.

```
/**
 * 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;
    });

```

```
}
```

Buat pekerjaan retensi kepatuhan.

```
/**
 * 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();
}
```

Buat pekerjaan penundaan hukum.

```

/**
 * 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();
    }

```

Buat pekerjaan retensi tata kelola baru.

```

/**
 * 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();
    }
}

```

- Untuk detail API, lihat [CreateJob](#) di Referensi AWS SDK for Java 2.x API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

```

- Untuk detail API, lihat [CreateJob](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.

```

- Untuk detail API, lihat [CreateJob](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteJobTagging** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `DeleteJobTagging`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
            });
    }

```

- Untuk detail API, lihat [DeleteJobTagging](#) di Referensi AWS SDK for Java 2.x API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

```

- Untuk detail API, lihat [DeleteJobTagging](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [DeleteJobTagging](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DescribeJob** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `DescribeJob`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)

CLI

AWS CLI

Untuk menggambarkan pekerjaan operasi batch Amazon S3

Berikut ini `describe-job` menyediakan parameter konfigurasi dan status untuk pekerjaan operasi batch yang ditentukan.

```
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"
  }
}

```

- Untuk detail API, lihat [DescribeJob](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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);
    });
}

```

- Untuk detail API, lihat [DescribeJob](#) di Referensi AWS SDK for Java 2.x API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

```

- Untuk detail API, lihat [DescribeJob](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.

```

- Untuk detail API, lihat [DescribeJob](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetJobTagging** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `GetJobTagging`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
        });
}
```

```
    });  
}
```

- Untuk detail API, lihat [GetJobTagging](#) di Referensi AWS SDK for Java 2.x API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
```

- Untuk detail API, lihat [GetJobTagging](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.

```

- Untuk detail API, lihat [GetJobTagging](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutJobTagging** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `PutJobTagging`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)

Java

SDK untuk Java 2.x

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
            });
    }

```

- Untuk detail API, lihat [PutJobTagging](#) di Referensi AWS SDK for Java 2.x API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

```

- Untuk detail API, lihat [PutJobTagging](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.

```

- Untuk detail API, lihat [PutJobTagging](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **UpdateJobPriority** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `UpdateJobPriority`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)

CLI

AWS CLI

Untuk memperbarui prioritas pekerjaan pekerjaan operasi batch Amazon S3

`update-job-priority` Contoh berikut memperbarui pekerjaan yang ditentukan ke prioritas baru.

```
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  
}
```

- Untuk detail API, lihat [UpdateJobPriority](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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;
}

```

- Untuk detail API, lihat [UpdateJobPriority](#) di Referensi AWS SDK for Java 2.x API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
```

- Untuk detail API, lihat [UpdateJobPriority](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.
```

- Untuk detail API, lihat [UpdateJobPriority](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **UpdateJobStatus** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan `UpdateJobStatus`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)

CLI

AWS CLI

Untuk memperbarui status pekerjaan operasi batch Amazon S3

`update-job-status` Contoh berikut membatalkan pekerjaan yang ditentukan yang menunggu persetujuan.

```
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"
}
```

`update-job-status` Contoh berikut mengonfirmasi dan menjalankan yang ditentukan yang menunggu persetujuan.

```
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"
}
```

update-job-statusContoh berikut membatalkan pekerjaan tertentu yang sedang berjalan.

```
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"
}
```

- Untuk detail API, lihat [UpdateJobStatus](#) di Referensi AWS CLI Perintah.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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
            });
    }

```

- Untuk detail API, lihat [UpdateJobStatus](#) di Referensi AWS SDK for Java 2.x API.

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

```

- Untuk detail API, lihat [UpdateJobStatus](#) di AWS SDK for Python (Boto3) Referensi API.

SAP ABAP

SDK for SAP ABAP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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.

```

- Untuk detail API, lihat [UpdateJobStatus](#) di AWS SDK untuk referensi SAP ABAP API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Contoh kode untuk Bucket Direktori S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan Bucket Direktori S3 dengan kit pengembangan AWS perangkat lunak (SDK).

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Ember Direktori S3

- [Contoh dasar untuk Bucket Direktori S3 menggunakan AWS SDK](#)
 - [Halo ember direktori Amazon S3](#)
 - [Pelajari dasar-dasar Bucket Direktori S3 dengan AWS SDK](#)
 - [Tindakan untuk Bucket Direktori S3 menggunakan AWS SDK](#)
 - [Gunakan AbortMultipartUploaddengan AWS SDK](#)
 - [Gunakan CompleteMultipartUploaddengan AWS SDK](#)
 - [Gunakan CopyObjectdengan AWS SDK](#)
 - [Gunakan CreateBucketdengan AWS SDK](#)
 - [Gunakan CreateMultipartUploaddengan AWS SDK](#)
 - [Gunakan CreateSessiondengan AWS SDK](#)
 - [Gunakan DeleteBucketdengan AWS SDK](#)
 - [Gunakan DeleteBucketEncryptiondengan AWS SDK](#)
 - [Gunakan DeleteBucketPolicydengan AWS SDK](#)
 - [Gunakan DeleteObjectdengan AWS SDK](#)
 - [Gunakan DeleteObjectsdengan AWS SDK](#)
 - [Gunakan GetBucketEncryptiondengan AWS SDK](#)
 - [Gunakan GetBucketPolicydengan AWS SDK](#)
 - [Gunakan GetObjectdengan AWS SDK](#)

- [Gunakan GetObjectAttributesdengan AWS SDK](#)
- [Gunakan HeadBucketdengan AWS SDK](#)
- [Gunakan HeadObjectdengan AWS SDK](#)
- [Gunakan ListDirectoryBucketsdengan AWS SDK](#)
- [Gunakan ListMultipartUploadsdengan AWS SDK](#)
- [Gunakan ListObjectsV2 dengan AWS SDK](#)
- [Gunakan ListPartsdengan AWS SDK](#)
- [Gunakan PutBucketEncryptiondengan AWS SDK](#)
- [Gunakan PutBucketPolicydengan AWS SDK](#)
- [Gunakan PutObjectdengan AWS SDK](#)
- [Gunakan UploadPartdengan AWS SDK](#)
- [Gunakan UploadPartCopydengan AWS SDK](#)
- [Skenario untuk Bucket Direktori S3 menggunakan AWS SDK](#)
 - [Buat URL presigned untuk bucket direktori Amazon S3 untuk mendapatkan objek menggunakan AWS SDK](#)

Contoh dasar untuk Bucket Direktori S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan dasar-dasar Bucket Direktori Amazon S3 dengan SDK. AWS

Contoh

- [Halo ember direktori Amazon S3](#)
- [Pelajari dasar-dasar Bucket Direktori S3 dengan AWS SDK](#)
- [Tindakan untuk Bucket Direktori S3 menggunakan AWS SDK](#)
 - [Gunakan AbortMultipartUploaddengan AWS SDK](#)
 - [Gunakan CompleteMultipartUploaddengan AWS SDK](#)
 - [Gunakan CopyObjectdengan AWS SDK](#)
 - [Gunakan CreateBucketdengan AWS SDK](#)
 - [Gunakan CreateMultipartUploaddengan AWS SDK](#)
 - [Gunakan CreateSessiondengan AWS SDK](#)
 - [Gunakan DeleteBucketdengan AWS SDK](#)

- [Gunakan DeleteBucketEncryptiondengan AWS SDK](#)
- [Gunakan DeleteBucketPolicydengan AWS SDK](#)
- [Gunakan DeleteObjectdengan AWS SDK](#)
- [Gunakan DeleteObjectsdengan AWS SDK](#)
- [Gunakan GetBucketEncryptiondengan AWS SDK](#)
- [Gunakan GetBucketPolicydengan AWS SDK](#)
- [Gunakan GetObjectdengan AWS SDK](#)
- [Gunakan GetObjectAttributesdengan AWS SDK](#)
- [Gunakan HeadBucketdengan AWS SDK](#)
- [Gunakan HeadObjectdengan AWS SDK](#)
- [Gunakan ListDirectoryBucketsdengan AWS SDK](#)
- [Gunakan ListMultipartUploadsdengan AWS SDK](#)
- [Gunakan ListObjectsV2 dengan AWS SDK](#)
- [Gunakan ListPartsdengan AWS SDK](#)
- [Gunakan PutBucketEncryptiondengan AWS SDK](#)
- [Gunakan PutBucketPolicydengan AWS SDK](#)
- [Gunakan PutObjectdengan AWS SDK](#)
- [Gunakan UploadPartdengan AWS SDK](#)
- [Gunakan UploadPartCopydengan AWS SDK](#)

Halo ember direktori Amazon S3

Contoh kode berikut menunjukkan cara memulai menggunakan bucket direktori Amazon S3.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di [Repositori Contoh Kode 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;
    }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .
- [CreateBucket](#)
- [ListDirectoryBuckets](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Pelajari dasar-dasar Bucket Direktori S3 dengan AWS SDK

Contoh-contoh kode berikut menunjukkan cara:

- Siapkan VPC dan VPC Endpoint.
- Siapkan Kebijakan, Peran, dan Pengguna untuk bekerja dengan bucket direktori S3 dan kelas penyimpanan S3 Express One Zone.
- Buat dua Klien S3.
- Buat dua bucket.
- Buat objek dan salin.
- Menunjukkan perbedaan kinerja.
- Isi ember untuk menunjukkan perbedaan leksikografis.

- Minta pengguna untuk melihat apakah mereka ingin membersihkan sumber daya.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Jalankan skenario interaktif yang mendemonstrasikan fitur 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 = createSessionAddObjects();

        // 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.deleteBucketAndObjectsAsync(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
            when object names are listed. 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 `ListObject` operations to
            behave
            differently, which can cause unexpected results. Let's add a few
            more
            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 createSessionAdd0bjects() {
        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 bucket0bject = "basic-text-object.txt";
        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, bucket0bject, "Look Ma, I'm a bucket!").join();
            s3DirectoriesActions.createSessionAsync(mS3ExpressClient,
mdirectoryBucketName).join();

            // Copy the object to the destination S3 bucket.
            s3DirectoriesActions.copyObjectAsync(mS3ExpressClient,
mregularBucketName, bucket0bject, mdirectoryBucketName, bucket0bject).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.");
        }
    }
}
```

```

    }
  }
}

```

Kelas pembungkus untuk metode Amazon S3 SDK.

```

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 {
            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.waitForVpcAvailable(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);
    });
}
}
}

```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Java 2.x .

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObject](#)
- [GetObject](#)
- [ListObjects](#)
- [PutObject](#)

PHP

SDK untuk PHP

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Jalankan skenario yang mendemonstrasikan dasar-dasar bucket direktori Amazon S3 dan 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;
    }
}
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk PHP .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)
 - [ListObjects](#)
 - [PutObject](#)

Python

SDK untuk Python (Boto3)

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Jalankan skenario yang mendemonstrasikan dasar-dasar bucket direktori Amazon S3 dan 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_yesno,
        ):
            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_yn):
            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"],
            )

```

Kelas pembungkus untuk fungsi Amazon S3 Express SDK.

```

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
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK untuk Python (Boto3).

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObject](#)
- [GetObject](#)
- [ListObjects](#)
- [PutObject](#)

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Tindakan untuk Bucket Direktori S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara melakukan tindakan Bucket Direktori S3 individual dengan AWS SDK. Setiap contoh menyertakan tautan ke GitHub, di mana Anda dapat menemukan instruksi untuk mengatur dan menjalankan kode.

Kutipan ini memanggil API Bucket Direktori S3 dan merupakan kutipan kode dari program yang lebih besar yang harus dijalankan dalam konteks. Anda dapat melihat tindakan dalam konteks di [Skenario untuk Bucket Direktori S3 menggunakan AWS SDK](#).

Contoh berikut hanya mencakup tindakan yang paling umum digunakan. Untuk daftar lengkapnya, lihat Referensi [API Bucket Direktori Amazon S3](#).

Contoh

- [Gunakan AbortMultipartUploaddengan AWS SDK](#)
- [Gunakan CompleteMultipartUploaddengan AWS SDK](#)
- [Gunakan CopyObjectdengan AWS SDK](#)
- [Gunakan CreateBucketdengan AWS SDK](#)
- [Gunakan CreateMultipartUploaddengan AWS SDK](#)
- [Gunakan CreateSessiondengan AWS SDK](#)
- [Gunakan DeleteBucketdengan AWS SDK](#)
- [Gunakan DeleteBucketEncryptiondengan AWS SDK](#)
- [Gunakan DeleteBucketPolicydengan AWS SDK](#)
- [Gunakan DeleteObjectdengan AWS SDK](#)
- [Gunakan DeleteObjectsdengan AWS SDK](#)
- [Gunakan GetBucketEncryptiondengan AWS SDK](#)
- [Gunakan GetBucketPolicydengan AWS SDK](#)
- [Gunakan GetObjectdengan AWS SDK](#)
- [Gunakan GetObjectAttributesdengan AWS SDK](#)
- [Gunakan HeadBucketdengan AWS SDK](#)
- [Gunakan HeadObjectdengan AWS SDK](#)
- [Gunakan ListDirectoryBucketsdengan AWS SDK](#)
- [Gunakan ListMultipartUploadsdengan AWS SDK](#)
- [Gunakan ListObjectsV2 dengan AWS SDK](#)
- [Gunakan ListPartsdengan AWS SDK](#)
- [Gunakan PutBucketEncryptiondengan AWS SDK](#)
- [Gunakan PutBucketPolicydengan AWS SDK](#)
- [Gunakan PutObjectdengan AWS SDK](#)
- [Gunakan UploadPartdengan AWS SDK](#)
- [Gunakan UploadPartCopydengan AWS SDK](#)

Gunakan **AbortMultipartUpload** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `AbortMultipartUpload`.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Batalkan unggahan multibagian dalam ember direktori.

```
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;
}
}

```

- Untuk detail API, lihat [AbortMultipartUpload](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **CompleteMultipartUpload** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan **CompleteMultipartUpload**.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Selesaikan unggahan multibagian dalam ember direktori.

```
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;
        }
    }
}

```

- Untuk detail API, lihat [CompleteMultipartUpload](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **CopyObject** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan **CopyObject**.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Salin objek dari bucket direktori ke bucket direktori.

```
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;
    }
}

```

- Untuk detail API, lihat [CopyObject](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **CreateBucket** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan **CreateBucket**.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat bucket direktori 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;
    }
}
```

- Untuk detail API, lihat [CreateBucket](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **CreateMultipartUpload** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan **CreateMultipartUpload**.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat unggahan multibagian dalam ember direktori.

```
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;
    }
}
```

- Untuk detail API, lihat [CreateMultipartUpload](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **CreateSession** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `CreateSession`.

Python

SDK untuk Python (Boto3)

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode 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

```

- Untuk detail API, lihat [CreateSession](#) di AWS SDK for Python (Boto3) Referensi API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteBucket** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `DeleteBucket`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)

Java

SDK untuk Java 2.x

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hapus bucket direktori 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;
    }
}
```

- Untuk detail API, lihat [DeleteBucket](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteBucketEncryption** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan **DeleteBucketEncryption**.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hapus konfigurasi enkripsi untuk bucket direktori.

```
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;
    }
}
```

- Untuk detail API, lihat [DeleteBucketEncryption](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteBucketPolicy** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `DeleteBucketPolicy`.

Java

SDK untuk Java 2.x

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Menghapus kebijakan bucket untuk bucket direktori.

```
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;
    }
}
```

- Untuk detail API, lihat [DeleteBucketPolicy](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteObject** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `DeleteObject`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hapus objek dalam ember direktori.

```
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;
        }
    }
}
```

- Untuk detail API, lihat [DeleteObject](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **DeleteObjects** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `DeleteObjects`.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hapus beberapa objek dalam ember direktori.

```
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;
    }
}
```

- Untuk detail API, lihat [DeleteObjects](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketEncryption** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan **GetBucketEncryption**.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Dapatkan konfigurasi enkripsi bucket direktori.

```
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.getServerSideEncryptionConfiguration().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;
}
}
```

- Untuk detail API, lihat [GetBucketEncryption](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetBucketPolicy** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `GetBucketPolicy`.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Dapatkan kebijakan bucket direktori.

```
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;
    }
}
```

- Untuk detail API, lihat [GetBucketPolicy](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetObject** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `GetObject`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)
- [Buat URL presigned untuk mendapatkan objek](#)

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Dapatkan objek dari ember direktori.

```
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;
    }
}
```

- Untuk detail API, lihat [GetObject](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **GetObjectAttributes** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `GetObjectAttributes`.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Dapatkan atribut objek dari bucket direktori.

```
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;
    }
}
```

- Untuk detail API, lihat [GetObjectAttributes](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **HeadBucket** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan **HeadBucket**.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Memeriksa apakah bucket direktori S3 yang ditentukan ada dan dapat diakses.

```
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;
    }
}
```

- Untuk detail API, lihat [HeadBucket](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **HeadObject** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `HeadObject`.

Java

SDK untuk Java 2.x

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Dapatkan metadata objek dalam ember direktori.

```
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;
    }
}
```

- Untuk detail API, lihat [HeadObject](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **ListDirectoryBuckets** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `ListDirectoryBuckets`.

Java

SDK untuk Java 2.x

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Daftar semua ember direktori.

```
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;
    }
}

```

- Untuk detail API, lihat [ListDirectoryBuckets](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **ListMultipartUploads** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `ListMultipartUploads`.

Java

SDK untuk Java 2.x

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat daftar unggahan multibagian dalam ember direktori.

```
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
    }
}
```

- Untuk detail API, lihat [ListMultipartUploads](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **ListObjectsV2** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `ListObjectsV2`.

Java

SDK untuk Java 2.x

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Daftar objek dalam ember direktori.

```
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;
    }
}
```

- Untuk detail API, lihat [ListObjectsV2](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **ListParts** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `ListParts`.

Java

SDK untuk Java 2.x

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat daftar bagian dari unggahan multibagian dalam bucket direktori.

```
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
    }
}
```

- Untuk detail API, lihat [ListParts](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutBucketEncryption** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan **PutBucketEncryption**.

Java

SDK untuk Java 2.x

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Setel enkripsi bucket ke bucket direktori.

```
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;
}
}
```

- Untuk detail API, lihat [PutBucketEncryption](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutBucketPolicy** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan **PutBucketPolicy**.

Java

SDK untuk Java 2.x

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Menerapkan kebijakan bucket ke bucket direktori.

```
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;
}
```

- Untuk detail API, lihat [PutBucketPolicy](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **PutObject** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `PutObject`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- [Pelajari dasar-dasarnya](#)

Java

SDK untuk Java 2.x

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Menempatkan objek ke dalam ember direktori.

```
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;
    }
}
```

- Untuk detail API, lihat [PutObject](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **UploadPart** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `UploadPart`.

Java

SDK untuk Java 2.x

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Unggah bagian dari unggahan multibagian untuk bucket direktori.

```
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;  
}
```

- Untuk detail API, lihat [UploadPart](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Gunakan **UploadPartCopy** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan `UploadPartCopy`.

Java

SDK untuk Java 2.x

 Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Buat salinan bagian berdasarkan ukuran objek sumber dan salin bagian individual ke ember direktori.

```
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;
}
```

- Untuk detail API, lihat [UploadPartCopy](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Skenario untuk Bucket Direktori S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menerapkan skenario umum di Bucket Direktori S3 dengan AWS SDK. Skenario ini menunjukkan kepada Anda cara menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam Bucket Direktori S3 atau digabungkan dengan yang lain. Layanan AWS Setiap skenario menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode.

Skenario menargetkan tingkat pengalaman menengah untuk membantu Anda memahami tindakan layanan dalam konteks.

Contoh

- [Buat URL presigned untuk bucket direktori Amazon S3 untuk mendapatkan objek menggunakan AWS SDK](#)

Buat URL presigned untuk bucket direktori Amazon S3 untuk mendapatkan objek menggunakan AWS SDK

Contoh kode berikut menunjukkan cara membuat URL presigned untuk bucket direktori S3 dan mendapatkan objek.

Java

SDK untuk Java 2.x

Note

Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di [Repositori Contoh Kode AWS](#).

Hasilkan URL GET yang telah ditetapkan sebelumnya untuk mengakses objek di bucket direktori 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.presignedGetObject(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;
    }
}
```

```
}
```

- Untuk detail API, lihat [GetObject](#) di Referensi AWS SDK for Java 2.x API.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihat [Mengembangkan dengan Amazon S3 menggunakan AWS SDK](#). Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Permintaan Otentikasi (AWS Versi Tanda Tangan 4)

Topik

- [Metode Autentikasi](#)
- [Pengantar Permintaan Penandatanganan](#)
- [Permintaan Otentikasi: Menggunakan Header Otorisasi \(AWS Versi Tanda Tangan 4\)](#)
- [Otentikasi Permintaan: Menggunakan Parameter Kueri \(AWS Versi Tanda Tangan 4\)](#)
- [Contoh: Perhitungan Tanda Tangan di AWS Tanda Tangan Versi 4](#)
- [Permintaan Otentikasi: Ung Browser-Based gah Menggunakan POST \(AWS Versi Tanda Tangan 4\)](#)
- [Kunci Kebijakan Khusus Otentikasi Tanda Tangan Amazon S3 Versi 4](#)

Setiap interaksi dengan Amazon S3 diautentikasi atau dilakukan secara anonim. Bagian ini menjelaskan otentikasi permintaan dengan algoritma AWS Signature Version 4.

Note

Jika Anda menggunakan AWS SDK (lihat Kode [Contoh dan Pustaka](#)) untuk mengirim permintaan, Anda tidak perlu membaca bagian ini karena klien SDK mengotentikasi permintaan Anda dengan menggunakan kunci akses yang Anda berikan. Kecuali Anda memiliki alasan bagus untuk tidak melakukannya, Anda harus selalu menggunakan AWS SDK. Di Wilayah yang mendukung kedua versi tanda tangan, Anda dapat meminta AWS SDK untuk menggunakan versi tanda tangan tertentu. Untuk informasi selengkapnya, lihat [Mengirim permintaan yang diautentikasi menggunakan AWS SDK](#). Anda perlu membaca bagian ini hanya jika Anda menerapkan algoritma AWS Signature Version 4 di klien kustom Anda.

Otentik AWS asi dengan Tanda Tangan Versi 4 menyediakan beberapa atau semua hal berikut, tergantung pada cara Anda memilih untuk menandatangani permintaan Anda:

- Verifikasi identitas pemohon — Permintaan yang diautentikasi memerlukan tanda tangan yang Anda buat dengan menggunakan kunci akses Anda (ID kunci akses, kunci akses rahasia). Untuk

informasi tentang mendapatkan kunci akses, lihat Mem [ahami dan Mendapatkan Kredensial Keamanan Anda](#) di Referensi Umum AWS. Jika Anda menggunakan kredentif keamanan sementara, perhitungan tanda tangan juga memerlukan token keamanan. Untuk informasi lebih lanjut, lihat [Meminta Kredensial Keamanan Sementara](#) dalam Panduan Pengguna IAM.

- In-transit perlindungan data — Untuk mencegah gangguan permintaan saat sedang transit, Anda menggunakan beberapa elemen permintaan untuk menghitung tanda tangan permintaan. Setelah menerima permintaan, Amazon S3 menghitung tanda tangan dengan menggunakan elemen permintaan yang sama. Jika komponen permintaan yang diterima oleh Amazon S3 tidak cocok dengan komponen yang digunakan untuk menghitung tanda tangan, Amazon S3 akan menolak permintaan tersebut.
- Melindungi dari penggunaan kembali bagian permintaan yang ditandatangani — Bagian permintaan yang ditandatangani (menggunakan AWS Tanda Tangan) valid dalam waktu 15 menit dari stempel waktu dalam permintaan. Pihak yang tidak berwenang yang memiliki akses ke permintaan yang ditandatangani dapat mengubah bagian permintaan yang tidak ditandatangani tanpa mempengaruhi validitas permintaan dalam jendela 15 menit. Karena itu, sebaiknya Anda memaksimalkan perlindungan dengan menandatangani header dan isi permintaan, membuat permintaan HTTPS ke Amazon S3, dan dengan menggunakan kunci `s3:x-amz-content-sha256` kondisi (lihat [Kunci Kebijakan Khusus Otentikasi Tanda Tangan Amazon S3 Versi 4](#)) dalam AWS kebijakan untuk mengharuskan pengguna menandatangani badan permintaan Amazon S3.

Note

Amazon S3 mendukung Signature Version 4, protokol untuk mengotentikasi permintaan API masuk ke AWS layanan, secara keseluruhan. Wilayah AWS Pada saat ini, Wilayah AWS dibuat sebelum 30 Januari 2014 akan terus mendukung protokol sebelumnya, Signature Version 2. Setiap Wilayah baru setelah 30 Januari 2014 hanya akan mendukung Tanda Tangan Versi 4 dan oleh karena itu semua permintaan ke Wilayah tersebut harus dibuat dengan Tanda Tangan Versi 4. Untuk informasi selengkapnya tentang AWS Tanda Tangan Versi 2, lihat [Menandatangani dan Mengotentikasi Permintaan RES](#) T di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Metode Autentikasi

Anda dapat mengungkapkan informasi otentikasi dengan menggunakan salah satu metode berikut:

- **Header Otorisasi HTTP** — Menggunakan `Authorization` header HTTP adalah metode paling umum untuk mengotentikasi permintaan Amazon S3. Semua operasi REST Amazon S3 (kecuali untuk unggahan berbasis browser menggunakan permintaan POST) memerlukan header ini. Untuk informasi selengkapnya tentang nilai `Authorization` header, dan cara menghitung tanda tangan dan opsi terkait, lihat [Permintaan Otentikasi: Menggunakan Header Otorisasi \(AWS Versi Tanda Tangan 4\)](#).
- **Parameter string kueri** — Anda dapat menggunakan string kueri untuk mengekspresikan permintaan sepenuhnya dalam URL. Dalam hal ini, Anda menggunakan parameter kueri untuk memberikan informasi permintaan, termasuk informasi otentikasi. Karena tanda tangan permintaan adalah bagian dari URL, jenis URL ini sering disebut sebagai URL yang ditandatangani sebelumnya. Anda dapat menggunakan URL yang telah ditandatangani sebelumnya untuk menyematkan tautan yang dapat diklik, yang dapat berlaku hingga tujuh hari, dalam HTML. Untuk informasi selengkapnya, lihat [Otentikasi Permintaan: Menggunakan Parameter Kueri \(AWS Versi Tanda Tangan 4\)](#).

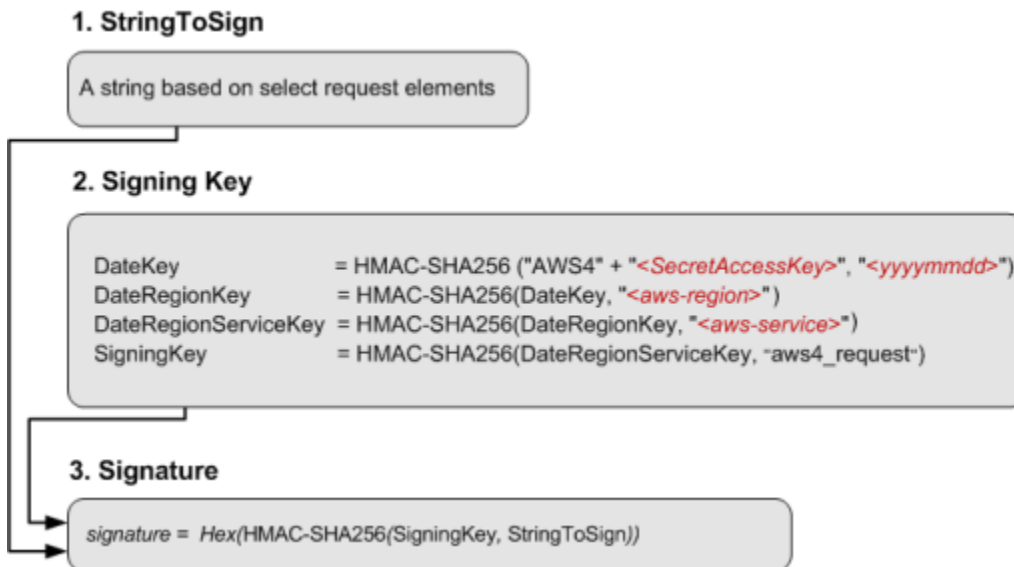
Amazon S3 juga mendukung upload berbasis browser yang menggunakan permintaan HTTP POST. Dengan permintaan HTTP POST, Anda dapat mengunggah konten ke Amazon S3 langsung dari browser. Untuk informasi tentang mengotentikasi permintaan POST, lihat [Browser-Based Unggah Menggunakan POST \(AWS Versi Tanda Tangan 4\)](#).

Pengantar Permintaan Penandatanganan

Informasi otentikasi yang Anda kirim dalam permintaan harus menyertakan tanda tangan. Untuk menghitung tanda tangan, pertama-tama Anda menggabungkan elemen permintaan pilih untuk membentuk string, yang disebut sebagai string yang akan ditandatangani. Anda kemudian menggunakan kunci penandatanganan untuk menghitung kode otentikasi pesan berbasis hash (HMAC) dari string yang akan ditandatangani.

Di AWS Signature Version 4, Anda tidak menggunakan kunci akses rahasia untuk menandatangani permintaan. Sebagai gantinya, pertama-tama Anda menggunakan kunci akses rahasia Anda untuk mendapatkan kunci penandatanganan. Kunci penandatanganan turunan khusus untuk tanggal, layanan, dan Wilayah. Untuk informasi selengkapnya tentang cara mendapatkan kunci penandatanganan dalam bahasa pemrograman yang berbeda, lihat [Contoh cara mendapatkan kunci penandatanganan untuk Tanda Tangan Versi 4](#).

Diagram berikut menggambarkan proses umum menghitung tanda tangan.



String yang akan ditandatangani tergantung pada jenis permintaan. Misalnya, saat Anda menggunakan header Otorisasi HTTP atau parameter kueri untuk otentikasi, Anda menggunakan berbagai kombinasi elemen permintaan untuk membuat string yang akan ditandatangani. Untuk permintaan HTTP POST, kebijakan POST dalam permintaan adalah string yang Anda tandatangani. Untuk informasi lebih lanjut tentang komputasi string untuk ditandatangani, ikuti tautan yang disediakan di akhir bagian ini.

Untuk kunci penandatanganan, diagram menunjukkan serangkaian perhitungan, di mana hasil dari setiap langkah Anda masukkan ke langkah berikutnya. Langkah terakhir adalah kunci penandatanganan.

Setelah menerima permintaan yang diautentikasi, server Amazon S3 membuat ulang tanda tangan dengan menggunakan informasi otentikasi yang terkandung dalam permintaan. Jika tanda tangan cocok, Amazon S3 memproses permintaan Anda; jika tidak, permintaan ditolak.

Untuk informasi selengkapnya tentang mengotentikasi permintaan, lihat topik berikut:

- [Permintaan Otentikasi: Menggunakan Header Otorisasi \(AWS Versi Tanda Tangan 4\)](#)
- [Otentikasi Permintaan: Menggunakan Parameter Kueri \(AWS Versi Tanda Tangan 4\)](#)
- [Browser-Based Unggah Menggunakan POST \(AWS Versi Tanda Tangan 4\)](#)

Permintaan Otentikasi: Menggunakan Header Otorisasi (AWS Versi Tanda Tangan 4)

Topik

- [Gambaran umum](#)
- [Perhitungan Tanda Tangan untuk Header Otorisasi: Mentransfer Payload dalam Satu Potongan \(AWS Versi Tanda Tangan 4\)](#)
- [Perhitungan Tanda Tangan untuk Header Otorisasi: Mentransfer Payload dalam Beberapa Potongan \(Unggahan Terpotong\) \(AWS Versi Tanda Tangan 4\)](#)
- [Perhitungan tanda tangan untuk header trailing \(unggahan berpotongan\) \(AWS Versi Tanda Tangan 4\)](#)

Gambaran umum

Menggunakan Authorization header HTTP adalah metode yang paling umum untuk memberikan informasi otentikasi. Kecuali [ali permintaan POST](#) dan permintaan yang ditandatangani menggunakan parameter kueri, semua operasi Amazon S3 menggunakan header Authorization permintaan untuk memberikan informasi otentikasi.

Berikut ini adalah contoh dari nilai Authorization header. Istirahat baris ditambahkan ke contoh ini untuk keterbacaan:

```
Authorization: AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,
SignedHeaders=host;range;x-amz-date,
Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024
```

Tabel berikut menjelaskan berbagai komponen dari nilai Authorization header dalam contoh sebelumnya:

Komponen	Deskripsi
AWS4-HMAC-SHA256	

Komponen	Deskripsi
	Algoritma yang digunakan untuk menghitung tanda tangan. Anda harus memberikan nilai ini saat menggunakan Tanda T AWS angan Versi 4 untuk otentikasi. String menentukan AWS Signature Version 4 (AWS4) dan algoritma penandatanganan (HMAC-SHA256).
Credential	ID kunci akses Anda dan informasi cakupan, yang mencakup tanggal, Wilayah, dan layanan yang digunakan untuk menghitung tanda tangan. String ini memiliki bentuk berikut: <pre data-bbox="602 825 1507 945"><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Di mana: • <i><date></i> nilai ditentukan menggunakan YYYYMMDD format. • <i><aws-service></i> nilai adalah s3 saat mengirim permintaan ke Amazon S3.
SignedHeaders	Daftar header permintaan yang dipisahkan titik koma yang Anda gunakan untuk menghitung. Signature Daftar ini hanya mencakup nama header, dan nama header harus dalam huruf kecil. Contoh: <pre data-bbox="602 1677 1507 1759">host;range;x-amz-date</pre>

Komponen	Deskripsi
Signature	Tanda tangan 256-bit dinyatakan sebagai 64 karakter heksadesimal huruf kecil. Contoh: <pre>fe5f80f77d5fa3beca038a248ff027d0445342fe2855d dc963176630326f1024</pre> Perhatikan bahwa perhitungan tanda tangan bervariasi tergantung pada opsi yang Anda pilih untuk mentransfer payload.

Perhitungan tanda tangan bervariasi tergantung pada metode yang Anda pilih untuk mentransfer payload permintaan. S3 mendukung opsi berikut:

- Transfer payload dalam satu potongan — Dalam hal ini, Anda memiliki opsi perhitungan tanda tangan berikut:
 - Opsi payload yang ditandatangani — Anda dapat secara opsional menghitung seluruh checksum payload dan memasukkannya ke dalam perhitungan tanda tangan. Ini memberikan keamanan tambahan tetapi Anda perlu membaca payload Anda dua kali atau menyimpannya dalam memori.

Misalnya, untuk mengunggah file, Anda perlu membaca file terlebih dahulu untuk menghitung hash payload untuk perhitungan tanda tangan dan sekali lagi untuk transmisi saat Anda membuat permintaan. Untuk muatan yang lebih kecil, pendekatan ini mungkin lebih disukai. Namun, untuk file besar, membaca file dua kali bisa tidak efisien, jadi Anda mungkin ingin mengunggah data dalam potongan sebagai gantinya.

Kami sarankan Anda menyertakan payload checksum untuk keamanan tambahan.

- Opsi payload yang tidak ditandatangani — Jangan sertakan checksum payload dalam perhitungan tanda tangan.

Untuk petunjuk langkah demi langkah untuk menghitung tanda tangan dan membangun nilai header Otorisasi, lihat [Perhitungan Tanda Tangan untuk Header Otorisasi: Mentransfer Payload dalam Satu Potongan \(AWS Versi Tanda Tangan 4\)](#).

- Transfer payload dalam beberapa potongan (unggahan potongan) — Dalam hal ini Anda mentransfer payload dalam potongan. Anda dapat mentransfer muatan dalam potongan terlepas dari ukuran payloadnya.

Anda dapat memecah muatan Anda menjadi potongan-potongan. Ini bisa berupa potongan tetap atau ukuran variabel. Dengan mengunggah data dalam potongan, Anda menghindari membaca seluruh payload untuk menghitung tanda tangan. Sebagai gantinya, untuk potongan pertama, Anda menghitung tanda tangan benih yang hanya menggunakan header permintaan. Potongan kedua berisi tanda tangan untuk potongan pertama, dan setiap potongan berikutnya berisi tanda tangan untuk potongan yang mendahuluinya. Di akhir unggahan, Anda mengirim potongan terakhir dengan 0 byte data yang berisi tanda tangan dari potongan terakhir muatan. Untuk informasi selengkapnya, lihat [Perhitungan Tanda Tangan untuk Header Otorisasi: Mentransfer Payload dalam Beberapa Potongan \(Unggahan Terpotong\) \(AWS Versi Tanda Tangan 4\)](#).

Saat menandatangani permintaan Anda, Anda dapat menggunakan Tanda Tangan AWS Versi 4 atau Tanda T AWS angan Versi 4A. Perbedaan utama antara keduanya ditentukan oleh bagaimana tanda tangan dihitung. Dengan AWS Tanda Tangan Versi 4A, tanda tangan tidak Region-specific menyertakan informasi dan dihitung menggunakan AWS4-ECDSA-P256-SHA256 algoritma.

Selain opsi ini, Anda memiliki opsi untuk menyertakan trailer dengan permintaan Anda. Untuk menyertakan trailer dengan permintaan Anda, Anda perlu menentukannya di header dengan mengatur nilai yang `x-amz-content-sha256` sesuai. Jika Anda menggunakan header trailing, Anda harus menyertakan `x-amz-trailer` dalam header dan menentukan nama header trailing sebagai string dalam daftar yang dipisahkan koma. Semua header trailing ditulis setelah potongan terakhir. Jika Anda mengunggah data dalam beberapa potongan, Anda harus mengirim potongan terakhir dengan 0 byte data sebelum mengirim header trailing.

Saat Anda mengirim permintaan, Anda harus memberi tahu Amazon S3 mana dari opsi sebelumnya yang telah Anda pilih dalam perhitungan tanda tangan Anda, dengan menambahkan `x-amz-content-sha256` header dengan salah satu nilai berikut:

Nilai header	Deskripsi
Nilai checksum muatan aktual	Nilai ini adalah checksum aktual dari objek Anda dan hanya mungkin ketika Anda mengunggah data dalam satu potongan.

Nilai header	Deskripsi
UNSIGNED-PAYLOAD	Gunakan ini saat Anda mengunggah objek sebagai potongan tunggal yang tidak ditandatangani.
STREAMING-UNSIGNED-PAYLOAD-TRAILER	Gunakan ini saat mengirim muatan yang tidak ditandatangani melalui beberapa potongan. Dalam hal ini Anda juga memiliki header trailing setelah potongan diunggah.
STREAMING-AWS4-HMAC-SHA256-PAYLOAD	Gunakan ini saat mengirim muatan melalui beberapa potongan, dan potongan ditandatangani menggunakan. AWS4-HMAC-SHA256 Ini menghasilkan tanda tangan SIGv4.
STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER	Gunakan ini saat mengirim muatan melalui beberapa potongan, dan potongan ditandatangani menggunakan. AWS4-HMAC-SHA256 Ini menghasilkan tanda tangan SIGv4. Selain itu, intisari untuk potongan disertakan sebagai header trailing.
STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD	Gunakan ini saat mengirim muatan melalui beberapa potongan, dan potongan ditandatangani menggunakan. AWS4-ECDSA-P256-SHA256 Ini menghasilkan tanda tangan Sigv4a.
STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD-TRAILER	Gunakan ini saat mengirim muatan melalui beberapa potongan, dan potongan ditandatangani menggunakan. AWS4-ECDSA-P256-SHA256 Ini menghasilkan tanda tangan Sigv4a. Selain itu, intisari untuk potongan disertakan sebagai header trailing.

Setelah menerima permintaan, Amazon S3 membuat ulang string untuk ditandatangani menggunakan informasi di `Authorization` header dan `date` header. Kemudian memverifikasi dengan layanan otentikasi tanda tangan cocok. Tanggal permintaan dapat ditentukan dengan menggunakan `HTTP Date` atau `x-amz-date` header. Jika kedua header hadir, diutamakan `x-amz-date` akan.

Jika tanda tangan cocok, Amazon S3 memproses permintaan Anda; jika tidak, permintaan Anda akan gagal.

Untuk informasi selengkapnya, lihat topik berikut:

[Perhitungan Tanda Tangan untuk Header Otorisasi: Mentransfer Payload dalam Satu Potongan \(AWS Versi Tanda Tangan 4\)](#)

[Perhitungan Tanda Tangan untuk Header Otorisasi: Mentransfer Payload dalam Beberapa Potongan \(Unggahan Terpotong\) \(AWS Versi Tanda Tangan 4\)](#)

[Perhitungan tanda tangan untuk header trailing \(unggahan berpotongan\) \(AWS Versi Tanda Tangan 4\)](#)

Perhitungan Tanda Tangan untuk Header Otorisasi: Mentransfer Payload dalam Satu Potongan (AWS Versi Tanda Tangan 4)

Saat menggunakan `Authorization` header untuk mengotentikasi permintaan, nilai header termasuk, antara lain, tanda tangan. Perhitungan tanda tangan bervariasi tergantung pada pilihan yang Anda buat untuk mentransfer payload ([Gambaran umum](#)). Bagian ini menjelaskan perhitungan tanda tangan saat Anda memilih untuk mentransfer payload dalam satu potongan. Bagian contoh (lihat [Contoh: Perhitungan Tanda Tangan](#)) menunjukkan perhitungan tanda tangan dan `Authorization` header hasil yang dapat Anda gunakan sebagai rangkaian pengujian untuk memverifikasi kode Anda.

Important

Saat mentransfer payload dalam satu potongan, Anda dapat secara opsional memilih untuk menyertakan hash payload dalam perhitungan tanda tangan, disebut sebagai payload yang ditandatangani (jika Anda tidak memasukkannya, payload dianggap tidak ditandatangani). Prosedur penandatanganan yang dibahas di bagian berikut berlaku untuk keduanya, tetapi perhatikan perbedaan berikut:

- Opsi payload yang ditandatangani — Anda menyertakan hash payload saat membangun permintaan kanonik (yang kemudian menjadi bagian dari `StringToSign`, seperti yang dijelaskan di bagian perhitungan tanda tangan). Anda juga menentukan nilai yang sama dengan nilai `x-amz-content-sha256` header saat mengirim permintaan ke S3.
- Opsi payload yang tidak ditandatangani — Anda menyertakan string literal `UNSIGNED-PAYLOAD` saat membuat permintaan kanonik, dan menetapkan nilai yang sama dengan nilai `x-amz-content-sha256` header saat mengirim permintaan ke Amazon S3.

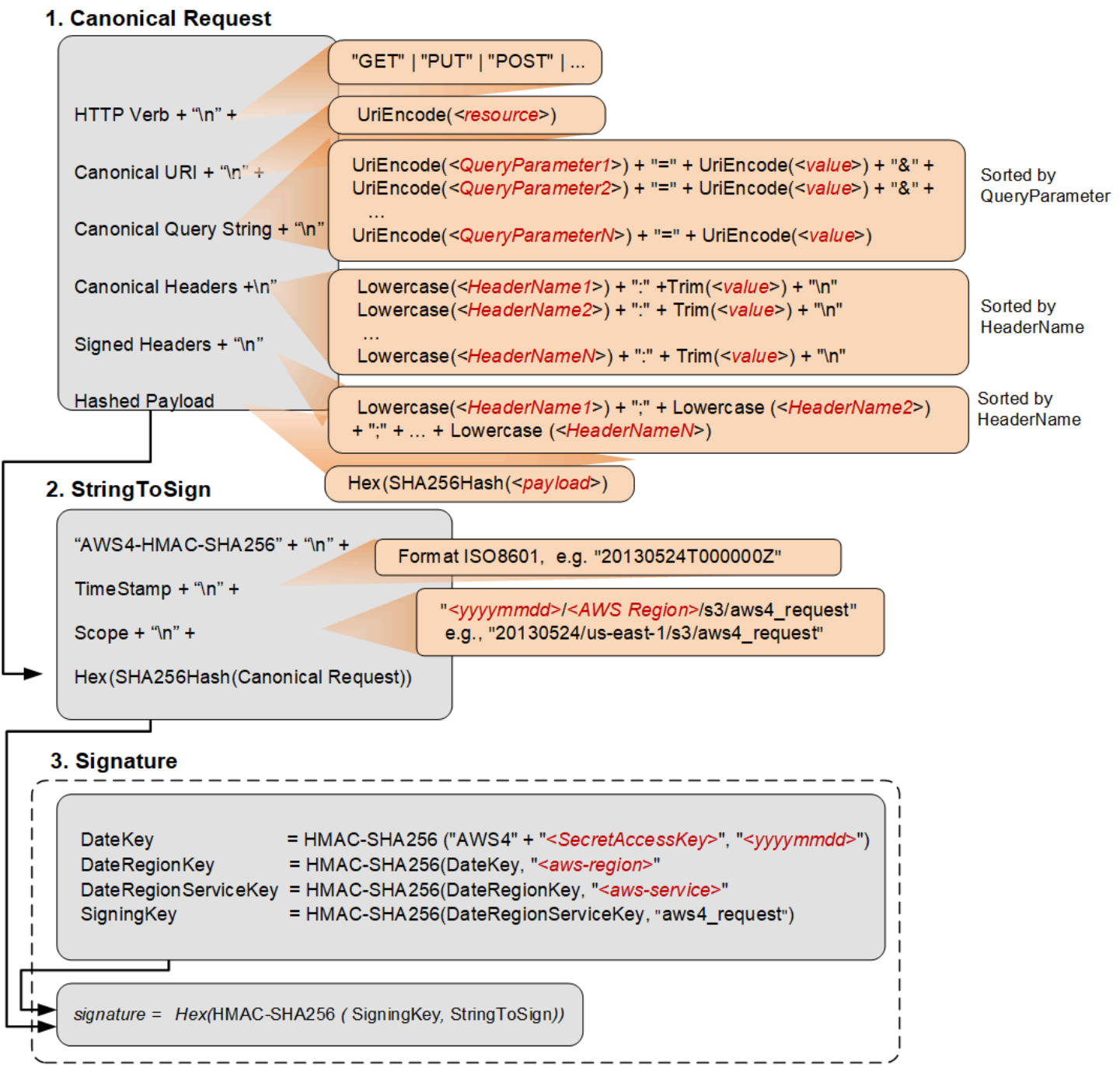
Saat Anda mengirim permintaan ke Amazon S3, nilai `x-amz-content-sha256` header memberi tahu Amazon S3 apakah payload ditandatangani atau tidak. Amazon S3 kemudian dapat membuat tanda tangan yang sesuai untuk verifikasi.

Dalam kedua kasus, karena nilai `x-amz-content-sha256` header sudah menjadi bagian dari `AndaHashedPayload`, Anda tidak diharuskan untuk menyertakan `x-amz-content-sha256` header sebagai header kanonik.

Menghitung Tanda Tangan

Untuk menghitung tanda tangan, pertama-tama Anda memerlukan string untuk ditandatangani. Anda kemudian menghitung HMAC-SHA256 hash string untuk ditandatangani dengan menggunakan kunci penandatanganan. Diagram berikut menggambarkan proses, termasuk berbagai komponen string yang Anda buat untuk penandatanganan.

Ketika Amazon S3 menerima permintaan yang diautentikasi, ia menghitung tanda tangan dan kemudian membandingkannya dengan tanda tangan yang Anda berikan dalam permintaan. Untuk alasan itu, Anda harus menghitung tanda tangan dengan menggunakan metode yang sama yang digunakan oleh Amazon S3. Proses menempatkan permintaan dalam bentuk yang disepakati untuk penandatanganan disebut kanonikalisasi.



Tabel berikut menjelaskan fungsi yang ditunjukkan dalam diagram. Anda perlu menerapkan kode untuk fungsi-fungsi ini.

Fungsi	Deskripsi
Lowercase ()	Ubah string menjadi huruf kecil.

Fungsi	Deskripsi
Hex()	Pengkodean huruf kecil 16.
SHA256Hash()	Fungsi hash kriptografi Secure Hash Algorithm (SHA).
HMAC-SHA256()	Menghitung HMAC dengan menggunakan algoritma SHA256 dengan kunci penandatanganan yang disediakan. Ini adalah tanda tangan terakhir.
Trim()	Hapus spasi kosong di depan atau di belakang.

Fungsi	Deskripsi
UriEncode()	URI mengkodekan setiap byte. UriEncode() harus menegakkan aturan berikut: • URI mengkodekan setiap byte kecuali karakter tanpa syarat: 'A'-'Z', 'a'-'z', '0'-'9', '-', '.', '_', dan '~'. • Karakter spasi adalah karakter yang dicadangkan dan harus dikodekan sebagai "%20" (dan bukan sebagai "+"). • Setiap byte yang dikodekan URI dibentuk oleh '%' dan nilai heksadesimal dua digit byte. • Huruf dalam nilai heksadesimal harus huruf besar, misalnya "%1A". • Enkode karakter garis miring ke depan, '/', di mana-mana kecuali dalam nama kunci objek. Misalnya, jika nama kunci objek adalah photos/Jan/sample.jpg , garis miring ke depan dalam nama kunci tidak dikodekan.  Important UriEncode Fungsi standar yang disediakan oleh platform pengembangan Anda mungkin tidak berfungsi karena perbedaan dalam implementasi dan ambiguitas terkait dalam RFC yang mendasarinya. Kami menyarankan Anda menulis UriEncode fungsi kustom Anda sendiri untuk memastikan bahwa pengkodean Anda akan berfungsi. Berikut ini adalah contoh fungsi UriEncode () di Java. <pre data-bbox="617 1644 1484 1875">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>

Fungsi	Deskripsi
	<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>

Tugas 1: Buat Permintaan Kanonik

Bagian ini memberikan gambaran umum tentang membuat permintaan kanonik.

Berikut ini adalah format permintaan kanonik yang digunakan Amazon S3 untuk menghitung tanda tangan. Agar tanda tangan cocok, Anda harus membuat permintaan kanonik dalam format ini:

```

<HTTPMethod>\n
<CanonicalURI>\n
<CanonicalQueryString>\n
<CanonicalHeaders>\n
<SignedHeaders>\n
<HashedPayload>

```

Di mana:

- **HTTPMethod** adalah salah satu metode HTTP, misalnya GET, PUT, HEAD, dan DELETE.
- **CanonicalURI** adalah URI-encoded versi komponen jalur absolut dari URI — semuanya dimulai dengan "/" yang mengikuti nama domain dan hingga akhir string atau ke karakter tanda tanya (?) jika Anda memiliki parameter string kueri. URI dalam contoh berikut, `/examplebucket/myphoto.jpg`, adalah jalur absolut dan Anda tidak mengkodekan "/" di jalur absolut:

```
http://s3.amazonaws.com/examplebucket/myphoto.jpg
```

Note

Anda tidak menormalkan jalur URI untuk permintaan ke Amazon S3. Misalnya, Anda mungkin memiliki bucket dengan objek bernama “my-object//example//photo.user”. Normalisasi jalur mengubah nama objek dalam permintaan menjadi “my-object/example/photo.user”. Ini adalah jalur yang salah untuk objek itu.

- **CanonicalQueryString** menentukan parameter URI-encoded string kueri. Anda URI-encode nama dan nilai satu per satu. Anda juga harus mengurutkan parameter dalam string kueri kanonik menurut abjad berdasarkan nama kunci. Penyortiran terjadi setelah pengkodean. String kueri dalam contoh URI berikut adalah `prefix=somePrefix&marker=someMarker&max-keys=20`:

```
http://s3.amazonaws.com/examplebucket?prefix=somePrefix&marker=someMarker&max-keys=20
```

String kueri kanonik adalah sebagai berikut (jeda baris ditambahkan ke contoh ini untuk keterbacaan):

```
UriEncode("marker")+"="+UriEncode("someMarker")+"&"+
UriEncode("max-keys")+"="+UriEncode("20") + "&" +
UriEncode("prefix")+"="+UriEncode("somePrefix")
```

Ketika permintaan menargetkan subresource, nilai parameter query yang sesuai akan menjadi string kosong (“”). Misalnya, URI berikut mengidentifikasi ACL subresource pada `examplebucket` bucket:

```
http://s3.amazonaws.com/examplebucket?acl
```

Dalam hal ini adalah sebagai berikut: CanonicalQueryString

```
UriEncode("acl") + "=" + ""
```

Jika URI tidak menyertakan '?', tidak ada string kueri dalam permintaan, dan Anda mengatur string kueri kanonik ke string kosong (""). Anda masih perlu memasukkan "\n".

- *CanonicalHeaders* adalah daftar header permintaan dengan nilainya. Nama header individu dan pasangan nilai dipisahkan oleh karakter baris baru ("\n"). Nama header harus dalam huruf kecil. Anda harus mengurutkan nama header menurut abjad untuk membangun string, seperti yang ditunjukkan dalam contoh berikut:

```
Lowercase(<HeaderName1>)+":"+Trim(<value>)+"\n"  
Lowercase(<HeaderName2>)+":"+Trim(<value>)+"\n"  
...  
Lowercase(<HeaderNameN>)+":"+Trim(<value>)+"\n"
```

Trim() Fungsi Lowercase() dan yang digunakan dalam contoh ini dijelaskan di bagian sebelumnya.

CanonicalHeaders Daftar harus mencakup yang berikut:

- hostheader HTTP.
- Jika Content-MD5 header ada dalam permintaan, Anda harus menambahkannya ke *CanonicalHeaders* daftar.
- Setiap x-amz-* header yang Anda rencanakan untuk disertakan dalam permintaan Anda juga harus ditambahkan. Misalnya, jika Anda menggunakan kredentif keamanan sementara, Anda harus menyertakan x-amz-security-token dalam permintaan Anda. Anda harus menambahkan header ini dalam daftar *CanonicalHeaders*.

Note

- x-amz-content-sha256Header diperlukan untuk semua permintaan AWS Signature Version 4. Ini menyediakan hash dari payload permintaan. Namun, Anda tidak diharuskan menyertakan x-amz-content-sha256 sebagai header kanonik karena S3 akan secara otomatis menggunakan nilainya saat menghitung hash payload yang dikirim dalam permintaan.
- Jika tidak ada payload, Anda harus memberikan hash dari string kosong.
- Jika Anda tidak ingin S3 memeriksa hash permintaan, Anda dapat menggunakan string literal sebagai gant "UNSIGNED-PAYLOAD" inya.

Berikut ini adalah contoh CanonicalHeaders string. Nama header dalam huruf kecil dan diurutkan.

Example Contoh 1

```
host:s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130708T220855Z
```

Example Contoh 2

```
host:s3.amazonaws.com
x-amz-content-sha256:UNSIGNED-PAYLOAD
x-amz-date:20130708T220855Z
```

Example Contoh 3

```
host:s3.amazonaws.com
x-amz-date:20130708T220855Z
```

Note

Untuk tujuan menghitung tanda tangan otorisasi, header host dan semua x-amz-* header, tidak termasuk x-amz-content-sha256, diperlukan. Namun, untuk mencegah gangguan data, Anda harus mempertimbangkan untuk memasukkan semua header dalam perhitungan tanda tangan.

Menandatangani x-amz-content-sha256 header bersifat opsional karena S3 akan menggunakan nilainya saat menghitung hash payload permintaan yang diterima.

- **SignedHeaders** adalah daftar nama header permintaan huruf kecil yang diurutkan berdasarkan abjad dan dipisahkan titik koma. Header permintaan dalam daftar adalah header yang sama yang Anda sertakan dalam string. CanonicalHeaders Dalam contoh sebelumnya, nilainya **SignedHeaders** adalah sebagai berikut:

Example Contoh 1 dan 2

```
host;x-amz-content-sha256;x-amz-date
```

Example Contoh 3

```
host;x-amz-date
```

- *HashedPayload* adalah nilai heksadesimal dari hash SHA256 dari payload permintaan.

```
Hex(SHA256Hash(<payload>))
```

Jika tidak ada payload dalam permintaan, Anda menghitung hash dari string kosong sebagai berikut:

```
Hex(SHA256Hash(""))
```

Hash mengembalikan nilai berikut:

```
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Misalnya, ketika Anda mengunggah objek dengan menggunakan permintaan PUT, Anda memberikan data objek dalam tubuh. Ketika Anda mengambil objek dengan menggunakan permintaan GET, Anda menghitung hash string kosong.

Tugas 2: Buat String untuk Ditandatangani

Bagian ini memberikan gambaran umum tentang membuat string untuk ditandatangani. Untuk petunjuk langkah demi langkah, lihat [Tugas 2: Membuat String untuk](#) Masuk Referensi Umum AWS.

String yang akan ditandatangani adalah gabungan dari string berikut:

```
"AWS4-HMAC-SHA256" + "\n" +  
timeStampISO8601Format + "\n" +  
<Scope> + "\n" +
```

```
Hex(SHA256Hash(<CanonicalRequest>))
```

String konstan AWS4-HMAC-SHA256 menentukan algoritma hash yang Anda gunakan, HMAC-SHA256. Ini `timeStamp` adalah waktu UTC saat ini dalam format ISO 8601 (misalnya, `20130524T000000Z`).

Scopemengikat tanda tangan yang dihasilkan ke tanggal tertentu Wilayah AWS, sebuah, dan layanan. Dengan demikian, tanda tangan yang dihasilkan hanya akan berfungsi di Wilayah tertentu dan untuk layanan tertentu. Tanda tangan berlaku selama tujuh hari setelah tanggal yang ditentukan.

```
date.Format(<YYYYMMDD>) + "/" + <region> + "/" + <service> + "/aws4_request"
```

Untuk Amazon S3, string layanan adalah `s3`. Untuk daftar `region` string, lihat [Wilayah dan Titik Akhir](#) di Referensi Umum AWS. Kolom Wilayah dalam tabel ini menyediakan daftar string Wilayah yang valid.

Cakupan berikut membatasi tanda tangan yang dihasilkan ke Wilayah `us-east-1` dan Amazon S3.

```
20130606/us-east-1/s3/aws4_request
```

Note

Scope harus menggunakan tanggal yang sama yang Anda gunakan untuk menghitung kunci penandatanganan, seperti yang dibahas di bagian berikut.

Tugas 3: Hitung Tanda Tangan

Di AWS Tanda Tangan Versi 4, alih-alih menggunakan kunci AWS akses untuk menandatangani permintaan, pertama-tama Anda membuat kunci penandatanganan yang dicakup ke Wilayah dan layanan tertentu. Untuk informasi selengkapnya tentang penandatanganan kunci, lihat [Pengantar Permintaan Penandatanganan](#).

```
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

Beberapa kasus penggunaan dapat memproses kunci tanda tangan hingga 7 hari. Untuk informasi selengkapnya, lihat [Berbagi Objek dengan Orang Lain](#).

Untuk daftar string Wilayah, lihat Wil [ayah dan Titik Akhir](#) di Referensi Umum AWS.

Menggunakan kunci penandatanganan memungkinkan Anda menyimpan AWS kredensialitas Anda di satu tempat yang aman. Misalnya, jika Anda memiliki beberapa server yang berkomunikasi dengan Amazon S3, Anda membagikan kunci penandatanganan dengan server tersebut; Anda tidak perlu menyimpan salinan kunci akses rahasia Anda di setiap server. Kunci penandatanganan berlaku hingga tujuh hari. Jadi setiap kali Anda menghitung kunci penandatanganan, Anda perlu membagikan kunci penandatanganan dengan server Anda. Untuk informasi selengkapnya, lihat [Permintaan Otentikasi \(AWS Versi Tanda Tangan 4\)](#).

Tanda tangan terakhir adalah HMAC-SHA256 hash dari string yang akan ditandatangani, menggunakan kunci penandatanganan sebagai kunci.

```
HMAC-SHA256(SigningKey, StringToSign)
```

Untuk petunjuk langkah demi langkah tentang membuat tanda tangan, lihat [Tugas 3: Membuat Tangan](#) Tangan di Referensi Umum AWS.

Contoh: Perhitungan Tanda Tangan

Anda dapat menggunakan contoh di bagian ini sebagai referensi untuk memeriksa perhitungan tanda tangan dalam kode Anda. Perhitungan yang ditunjukkan dalam contoh menggunakan data berikut:

- Contoh kunci akses.

Parameter	Nilai
AWSSecretAccessKey	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

- Minta stempel waktu 20130524T000000Z (). Fri, 24 May 2013 00:00:00 GMT
- Nama bucket `examplebucket`.
- Ember diasumsikan berada di Wilayah Timur AS (Virginia Utara). Kredensi Scope dan Signing Key perhitungan digunakan `us-east-1` sebagai penentu Wilayah. Untuk informasi tentang Wilayah lain, lihat [Wilayah dan Titik Akhir](#) di Referensi Umum AWS.
- Anda dapat menggunakan permintaan gaya path atau virtual hosted—style. Contoh berikut menunjukkan cara menandatangani permintaan gaya host virtual, misalnya:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

Untuk informasi selengkapnya, lihat Hosting [Virtual Bucket](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Contoh: GET Object

Contoh berikut mendapatkan 10 byte pertama dari sebuah objek (`test.txt`) dari `examplebucket`. Untuk informasi selengkapnya tentang tindakan API, lihat [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
```

Karena permintaan GET ini tidak menyediakan konten tubuh apa pun, `x-amz-content-sha256` nilainya dapat berupa hash dari badan permintaan kosong atau string literal `"UNSIGNED-PAYLOAD"`. Langkah-langkah berikut menunjukkan perhitungan tanda tangan dan konstruksi Authorization header menggunakan hash dari string kosong.

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

```

Dalam string permintaan kanonik, baris terakhir adalah hash dari badan permintaan kosong. Baris ketiga kosong karena tidak ada parameter kueri dalam permintaan.

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. Tanda tangan

```

f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41

```

4. Header otorisasi

AuthorizationHeader yang dihasilkan adalah sebagai berikut:

```

AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;range;x-amz-content-sha256;x-amz-
date,Signature=f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41

```

Contoh: Objek PUT

Contoh permintaan PUT ini membuat objek (test\$file.text) diexamplebucket. Contoh tersebut mengasumsikan sebagai berikut:

- Anda meminta REDUCED_REDUNDANCY sebagai kelas penyimpanan dengan menambahkan header x-amz-storage-class permintaan. Untuk informasi tentang kelas penyimpanan, lihat Kelas [Penyimpanan](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.
- Isi file yang diunggah adalah string, "Welcome to Amazon S3." x-amz-content-sha256 Nilai permintaan didasarkan pada string ini.

Untuk informasi tentang tindakan API, lihat [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>
```

Langkah-langkah berikut menunjukkan perhitungan tanda tangan.

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
```

Dalam permintaan kanonik, baris ketiga kosong karena tidak ada parameter kueri dalam permintaan. Header `x-amz-content-sha256` kanonik dapat secara opsional ditandatangani karena hash payloadnya sudah disediakan di bagian bawah permintaan. Baris terakhir adalah hash tubuh, yang harus sama dengan `x-amz-content-sha256` header nilai yang dikirim ke S3 dalam permintaan 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. Tanda tangan

```
98ad721746da40c64f1a55b78f14c238d841ea1380cd77a1b5971af0ece108bd
```

4. Header otorisasi

AuthorizationHeader yang dihasilkan adalah sebagai berikut:

```
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
```

Contoh: GET Bucket Lifecycle

Permintaan GET berikut mengambil konfigurasi siklus hidup. `examplebucket` Untuk informasi tentang tindakan API, lihat [GetBucketLifecycleConfiguration](#).

```
GET ?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Karena permintaan tidak menyediakan konten tubuh apa pun, nilai `x-amz-content-sha256` header adalah hash dari badan permintaan kosong. Langkah-langkah berikut menunjukkan perhitungan tanda tangan.

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
```

Dalam permintaan kanonik, baris terakhir adalah hash dari badan permintaan kosong.

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. Tanda tangan

```
fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

4. Header otorisasi

AuthorizationHeader yang dihasilkan adalah sebagai berikut:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/  
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-  
date,Signature=fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

Contoh: Dapatkan Bucket (Daftar Objek)

Contoh berikut mengambil daftar objek dari examplebucket bucket. Untuk informasi tentang tindakan API, lihat [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
```

Karena permintaan tidak menyediakan badan, nilainya x-amz-content-sha256 adalah hash dari badan permintaan kosong. Langkah-langkah berikut menunjukkan perhitungan tanda tangan.

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
```

Dalam string kanonik, baris terakhir adalah hash dari badan permintaan kosong.

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. Tanda tangan

```
34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```

4. Header otorisasi

AuthorizationHeader yang dihasilkan adalah sebagai berikut:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-
date,Signature=34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```


Perhitungan Tanda Tangan untuk Header Otorisasi: Mentransfer Payload dalam Beberapa Potongan (Unggahan Terpotong) (AWS Versi Tanda Tangan 4)

Seperti dijelaskan dalam [Gambaran umum](#), saat mengotentikasi permintaan menggunakan Authorization header, Anda memiliki opsi untuk mengunggah payload dalam potongan. Anda dapat mengirim data dalam ukuran tetap atau potongan ukuran variabel. Bagian ini menjelaskan proses perhitungan tanda tangan dalam unggahan potongan, cara Anda membuat badan potongan, dan cara kerja penandatanganan tertunda di mana Anda pertama kali mengunggah potongan, dan mengirim tanda tangannya di bagian berikutnya. Bagian contoh (lihat [Contoh: Objek PUT](#)) menunjukkan perhitungan tanda tangan dan Authorization header hasil yang dapat Anda gunakan sebagai rangkaian pengujian untuk memverifikasi kode Anda.

Note

Saat mentransfer data dalam serangkaian potongan, Anda harus melakukan salah satu dari hal berikut:

- Secara eksplisit menentukan panjang konten total (panjang objek dalam byte ditambah metadata di setiap potongan) menggunakan header Content-Length HTTP. Untuk melakukan ini, Anda harus menghitung terlebih dahulu panjang total payload, termasuk metadata yang Anda kirim di setiap potongan, sebelum memulai permintaan Anda.
- Tentukan header Transfer-Encoding HTTP. Jika Anda menyertakan Transfer-Encoding header dan menentukan nilai apa pun selain identity, Anda harus menghilangkan Content-Length header.

Untuk semua permintaan, Anda harus menyertakan x-amz-decoded-content-length header, menentukan ukuran objek dalam byte.

Setiap perhitungan tanda tangan potongan mencakup tanda tangan dari potongan sebelumnya. Untuk memulai, Anda membuat tanda tangan benih hanya menggunakan header. Anda menggunakan tanda tangan benih dalam perhitungan tanda tangan dari potongan pertama. Untuk setiap potongan berikutnya, Anda membuat tanda tangan potongan yang menyertakan tanda tangan dari potongan sebelumnya. Dengan demikian, tanda tangan potongan dirantai bersama; yaitu, tanda

tangan dari potongan n adalah fungsi F (chunk n , signature (chunk $n-1$)). Rangkaian memastikan bahwa Anda mengirim potongan dalam urutan yang benar.

Untuk melakukan unggahan dalam potongan, lakukan hal berikut:

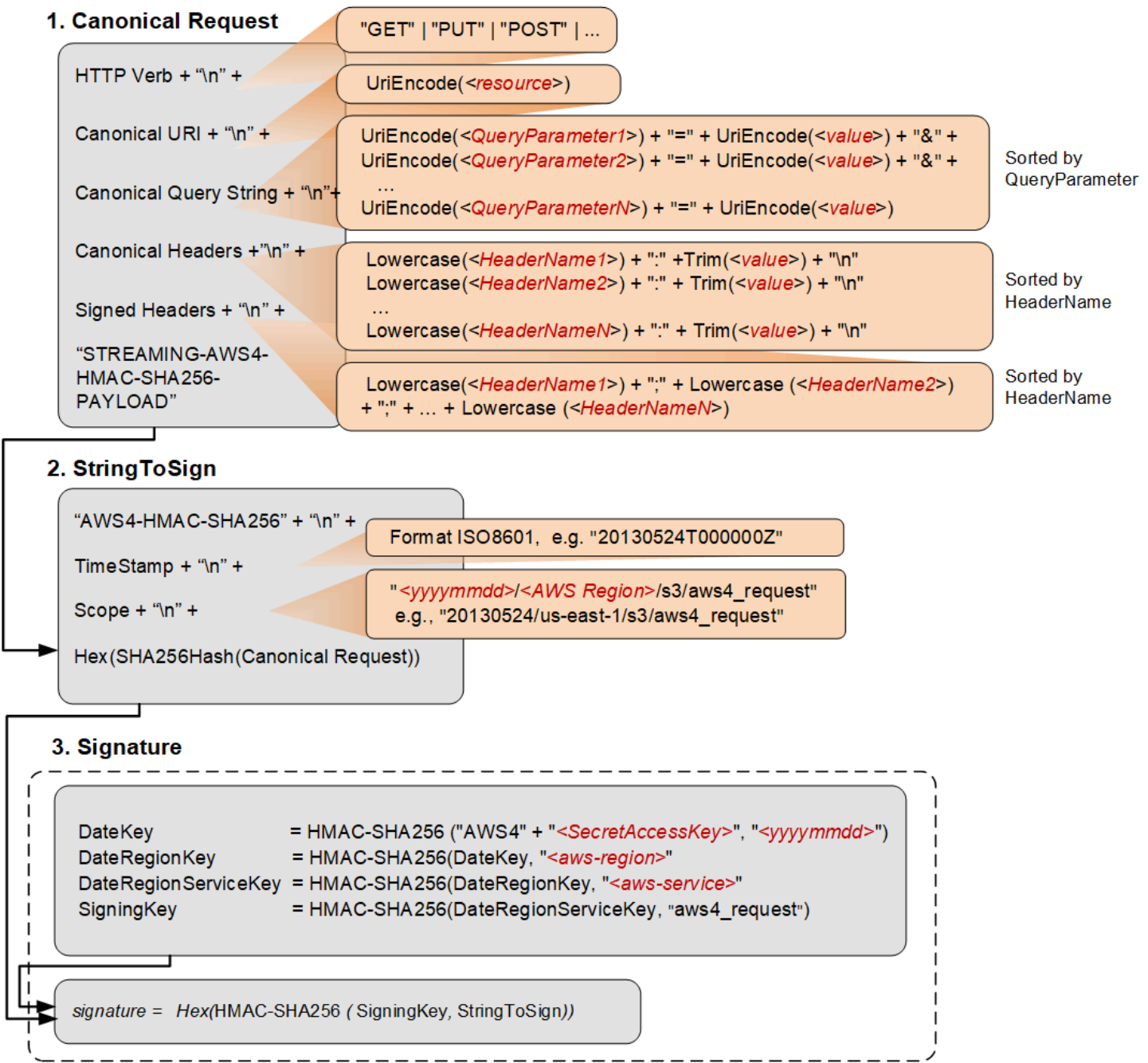
1. Tentukan ukuran potongan muatan. Anda membutuhkan ini ketika Anda menulis kode.

Ukuran potongan harus minimal 8 KB. Kami merekomendasikan ukuran potongan minimal 64 KB untuk kinerja yang lebih baik. Ukuran potongan ini berlaku untuk semua potongan kecuali yang terakhir. Potongan terakhir yang Anda kirim bisa lebih kecil dari 8 KB. Jika muatan Anda kecil dan dapat masuk ke dalam satu potongan, maka itu bisa lebih kecil dari 8 KB.

2. Buat tanda tangan benih untuk dimasukkan dalam potongan pertama. Untuk informasi selengkapnya, lihat [Menghitung Tanda Tangan Benih](#).
3. Buat potongan pertama dan streaming. Untuk informasi selengkapnya, lihat [Mendefinisikan Tubuh Potongan](#).
4. Untuk setiap potongan berikutnya, hitung tanda tangan potongan yang menyertakan tanda tangan sebelumnya dalam string yang Anda tandatangani, buat potongan, dan kirimkan. Untuk informasi selengkapnya, lihat [Mendefinisikan Tubuh Potongan](#).
5. Kirim potongan tambahan terakhir, yang sama dengan potongan lain dalam konstruksi, tetapi memiliki nol byte data. Untuk informasi selengkapnya, lihat [Mendefinisikan Tubuh Potongan](#).

Menghitung Tanda Tangan Benih

Diagram berikut menggambarkan proses penghitungan tanda tangan benih.



Tabel berikut menjelaskan fungsi yang ditunjukkan dalam diagram. Anda perlu menerapkan kode untuk fungsi-fungsi ini.

Fungsi	Deskripsi
Lowercase()	Ubah string menjadi huruf kecil.
Hex()	Pengkodean huruf kecil 16.

Fungsi	Deskripsi
SHA256Hash()	Fungsi hash kriptografi Secure Hash Algorithm (SHA).
HMAC-SHA256()	Menghitung HMAC dengan menggunakan algoritma SHA256 dengan kunci penandatanganan yang disediakan. Ini adalah tanda tangan terakhir.
Trim()	Hapus spasi kosong di depan atau di belakang.

Fungsi	Deskripsi
<code>UriEncode()</code>	URI mengkodekan setiap byte. <code>UriEncode()</code> harus menegakkan aturan berikut: • URI mengkodekan setiap byte kecuali karakter tanpa syarat: 'A'-'Z', 'a'-'z', '0'-'9', '-', '.', '_', dan '~'. • Karakter spasi adalah karakter yang dicadangkan dan harus dikodekan sebagai "%20" (dan bukan sebagai "+"). • Setiap byte yang dikodekan URI dibentuk oleh '%' dan nilai heksadesimal dua digit byte. • Huruf dalam nilai heksadesimal harus huruf besar, misalnya "%1A". • Enkode karakter garis miring ke depan, '/', di mana-mana kecuali dalam nama kunci objek. Misalnya, jika nama kunci objek adalah <code>photos/Jan/sample.jpg</code>, garis miring ke depan dalam nama kunci tidak dikodekan.  Important <code>UriEncode</code> Fungsi standar yang disediakan oleh platform pengembangan Anda mungkin tidak berfungsi karena perbedaan dalam implementasi dan ambiguitas terkait dalam RFC yang mendasarinya. Kami menyarankan Anda menulis <code>UriEncode</code> fungsi kustom Anda sendiri untuk memastikan bahwa pengkodean Anda akan berfungsi. Berikut ini adalah contoh fungsi <code>UriEncode()</code> di Java. <pre>public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBuil lder(); for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i);</pre>

Fungsi	Deskripsi
	<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>

Untuk informasi tentang proses penandatanganan, lihat [Perhitungan Tanda Tangan untuk Header Otorisasi: Mentransfer Payload dalam Satu Potongan \(AWS Versi Tanda Tangan 4\)](#). Prosesnya sama, kecuali bahwa penciptaan CanonicalRequest berbeda sebagai berikut:

- Selain header permintaan yang ingin Anda tambahkan, Anda harus menyertakan header berikut:

Header	Deskripsi
x-amz-content-sha256	Header ini diperlukan untuk semua permintaan AWS Signature Version 4. Tetapkan nilai STREAMING-AWS4-HMAC-SHA256-PAYLOAD untuk menunjukkan bahwa tanda tangan hanya mencakup header dan tidak ada payload.

Header	Deskripsi
Content-Encoding	Atur nilai ke <code>aws-chunked</code> . Amazon S3 mendukung beberapa nilai pengkodean konten. Anda dapat menentukan pengkodean konten kustom Anda saat menggunakan API streaming Signature Versi 4. Contoh: Content-Encoding : <code>aws-chunked, gzip</code> Amazon S3 menyimpan objek yang dihasilkan tanpa <code>aws-chunked</code> nilai di <code>content-encoding</code> header. Jika <code>aws-chunked</code> merupakan satu-satunya nilai yang Anda berikan di <code>content-encoding</code> header, S3 menganggap <code>content-encoding</code> header kosong dan tidak mengembalikan header ini saat Anda mengambil objek.
x-amz-decoded-content-length	Tetapkan nilai ke panjang, dalam byte, dari data yang akan dipotong, tanpa menghitung metadata apa pun. Misalnya, jika Anda mengunggah file 4 GB, atur nilainya ke 4294967296. Ini adalah ukuran mentah dari objek yang akan diunggah (data yang ingin Anda simpan di Amazon S3).

Header	Deskripsi
Content-Length	Tetapkan nilai ke ukuran aktual dari badan HTTP yang ditransmisikan, yang mencakup panjang data Anda (nilai yang ditetapkan untuk <code>x-amz-decoded-content-length</code>), ditambah metadata potongan. Setiap potongan memiliki metadata, seperti tanda tangan dari potongan sebelumnya. Perhitungan potongan dibahas di bagian berikut. Jika Anda menyertakan <code>Transfer-Encoding</code> header dan menentukan nilai apa pun selain <code>identity</code> , Anda tidak boleh menyertakan <code>Content-Length</code> header.

Anda mengirim potongan pertama dengan tanda tangan benih. Anda harus membangun potongan seperti yang dijelaskan di bagian berikut.

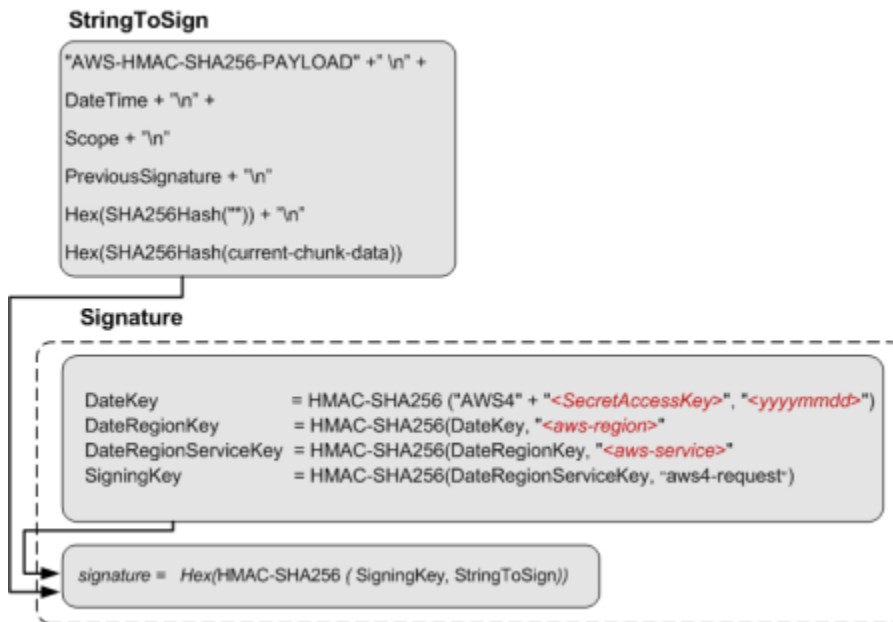
Mendefinisikan Tubuh Potongan

Semua potongan menyertakan beberapa metadata. Setiap potongan harus sesuai dengan struktur berikut:

```
string(IntHexBase(chunk-size)) + ";chunk-signature=" + signature + \r\n + chunk-data + \r\n
```

Di mana:

- `IntHexBase()` adalah fungsi yang Anda tulis untuk mengubah ukuran potongan integer menjadi heksadesimal. Misalnya, jika ukuran potongan adalah 65536, string heksadesimal adalah "10000".
- *chunk-size* adalah ukuran, dalam byte, dari chunk-data, tanpa metadata. Misalnya, jika Anda mengunggah objek 65 KB dan menggunakan ukuran potongan 64 KB, Anda mengunggah data dalam tiga potongan: yang pertama akan menjadi 64 KB, yang kedua 1 KB, dan potongan terakhir dengan 0 byte.
- *signature* Untuk setiap potongan, Anda menghitung tanda tangan menggunakan string berikut untuk ditandatangani. Untuk potongan pertama, Anda menggunakan seed-signature sebagai tanda tangan sebelumnya.



Ukuran data potongan akhir yang Anda kirim adalah 0, meskipun tubuh potongan masih berisi metadata, termasuk tanda tangan dari potongan sebelumnya.

Contoh: Objek PUT

Anda dapat menggunakan contoh di bagian ini sebagai referensi untuk memeriksa perhitungan tanda tangan dalam kode Anda. Sebelum Anda meninjau contoh, perhatikan hal berikut:

- Perhitungan tanda tangan dalam contoh ini menggunakan contoh kredentif keamanan berikut.

Parameter	Nilai
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

- Semua contoh menggunakan stempel waktu permintaan 20130524T000000Z (). Fri, 24 May 2013 00:00:00 GMT
- Semua contoh digunakan examplebucket sebagai nama bucket.

- Bucket diasumsikan berada di Wilayah AS Timur (Virginia Utara), dan kredensial serta Scope Signing Key perhitungan digunakan us-east-1 sebagai penentu Wilayah. Untuk informasi selengkapnya, lihat [Wilayah dan Titik Akhir](#) dalam Referensi Umum Amazon Web Services.
- Anda dapat menggunakan gaya jalur atau permintaan gaya yang dihosting virtual. Contoh berikut menggunakan permintaan gaya yang dihosting virtual, misalnya:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

Untuk informasi selengkapnya, lihat Hosting [Virtual Bucket](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Contoh berikut mengirimkan PUT permintaan untuk mengunggah objek. Perhitungan tanda tangan mengasumsikan sebagai berikut:

- Anda mengunggah file teks 65 KB, dan konten file adalah string satu karakter yang terdiri dari huruf 'a'.
- Ukuran potongan adalah 64 KB. Akibatnya, payload diunggah dalam tiga potongan, 64 KB, 1 KB, dan potongan terakhir dengan 0 byte data potongan.
- Objek yang dihasilkan memiliki nama kuncichunk0object.txt.
- Anda meminta REDUCED_REDUNDANCY sebagai kelas penyimpanan dengan menambahkan header x-amz-storage-class permintaan.

Untuk informasi tentang tindakan API, lihat [PutObject](#). Sintaks permintaan umum adalah sebagai berikut:

```
PUT /examplebucket/chunk0object.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>
```

Langkah-langkah berikut menunjukkan perhitungan tanda tangan.

1. Tanda tangan benih - Buat String untuk Ditandatangani

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
```

Dalam permintaan kanonik, baris ketiga kosong karena tidak ada parameter kueri dalam permintaan. Baris terakhir adalah string konstan yang disediakan sebagai nilai Payload yang di-hash, yang harus sama dengan nilai. `x-amz-content-sha256` header

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
cee3fed04b70f867d036f722359b0b1f2f0e5dc0efadbc082b76c4c60e316455
```

Note

Untuk informasi tentang setiap baris dalam string yang akan ditandatangani, lihat diagram yang menjelaskan perhitungan tanda tangan benih.

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +  
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Tanda Tangan Benih

```
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
```

4. Header otorisasi

Header Otorisasi yang dihasilkan adalah sebagai berikut:

```
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. Potongan 1: (65536 byte, dengan nilai 97 untuk huruf 'a')

a. Potongan string untuk ditandatangani:

```
AWS4-HMAC-SHA256-PAYLOAD  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9  
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855  
bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

Note

Untuk informasi tentang setiap baris dalam string yang akan ditandatangani, lihat diagram sebelumnya yang menunjukkan berbagai komponen string yang akan ditandatangani (misalnya, tiga baris terakhir adalah, `previous-signature`, `hash("")`, dan `hash(current-chunk-data)`).

b. Tanda tangan potongan:

```
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
```

c. Data potongan dikirim:

```
10000; chunk-  
signature=ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648  
<65536-bytes>
```

6. Potongan 2: (1024 byte, dengan nilai 97 untuk huruf 'a')

a. Potongan string untuk ditandatangani:

```
AWS4-HMAC-SHA256-PAYLOAD  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648  
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855  
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. Tanda tangan potongan:

```
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497
```

c. Data potongan dikirim:

```
400; chunk-  
signature=0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497  
<1024 bytes>
```

7. Potongan 3: (0 byte data)

a. Potongan string untuk ditandatangani:

```
AWS4-HMAC-SHA256-PAYLOAD  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497  
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855  
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```


b. Tanda tangan potongan:

```
b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

c. Data potongan dikirim:

```
0; chunk-  
signature=b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

Perhitungan tanda tangan untuk header trailing (unggahan berpotongan) (AWS Versi Tanda Tangan 4)

Saat mengotentikasi permintaan menggunakan Authorization header, Anda juga dapat mengunggah payload dalam potongan. Saat Anda mengirim data untuk objek dalam potongan, Anda juga memiliki opsi untuk menyertakan header trailing. (Untuk informasi selengkapnya, lihat [Perhitungan Tanda Tangan untuk Header Otorisasi: Mentransfer Payload dalam Beberapa Potongan \(Unggahan Terpotong\) \(AWS Versi Tanda Tangan 4\)](#).) Bagian ini menjelaskan langkah-langkah yang perlu Anda ambil ketika Anda ingin menyertakan header trailing di akhir upload beberapa chunk Anda.

Important

Ketika Anda menyertakan header trailing, Anda harus mengirim yang berikut di header awal Anda:

- Anda `x-amz-content-sha256` harus mengatur nilai yang sesuai yang menunjukkan trailer akan disertakan. Untuk melihat nilai yang dapat diterima `x-amz-content-sha256`, lihat [Permintaan Otentikasi: Menggunakan Header Otorisasi \(AWS Versi Tanda Tangan 4\)](#).
- Anda harus mengatur `x-amz-trailer` untuk menunjukkan konten yang Anda sertakan di header trailing Anda.

Header trailing hanya dikirim setelah potongan telah diunggah. Potongan sebelumnya dikirim seperti biasa dan ditandatangani seperti yang dijelaskan di bagian sebelumnya, termasuk mengirim potongan terakhir dengan muatan 0 byte. Header trailing disertakan sebagai potongan mereka sendiri dan dikirim setelah potongan terakhir dengan muatan 0 byte. Misalnya, jika data Anda diakhiri dengan potongan 100 KB, Anda akan mengirim yang berikut:

- Potongan data sebelumnya
- 100 KB potongan akhir objek
- Potongan 0 byte menandakan akhir objek
- Potongan header yang tertinggal

Contoh: Memeriksa perhitungan tanda tangan

Anda dapat menggunakan contoh di bagian ini sebagai referensi untuk memeriksa perhitungan tanda tangan dalam kode Anda. Sebelum Anda meninjau contoh, perhatikan hal berikut:

- Perhitungan tanda tangan dalam contoh ini menggunakan contoh kredentif keamanan berikut.

Parameter	Nilai
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecret AccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

- Semua contoh menggunakan stempel waktu permintaan 20130524T000000Z (). Fri, 24 May 2013 00:00:00 GMT
- Semua contoh digunakan examplebucket sebagai nama bucket.
- Bucket diasumsikan berada di Wilayah AS Timur (Virginia Utara), dan kredensial serta Scope Signing Key perhitungan digunakan us-east-1 sebagai penentu Wilayah. Untuk informasi selengkapnya, lihat [Wilayah dan titik akhir](#) dalam Referensi Umum Amazon Web Services.
- Anda dapat menggunakan gaya jalur atau permintaan gaya yang dihosting virtual. Contoh berikut menggunakan permintaan gaya yang dihosting virtual, misalnya:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

Untuk informasi selengkapnya, lihat Hosting [Virtual Bucket](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Contoh berikut mengirimkan PUT permintaan untuk mengunggah objek. Perhitungan tanda tangan mengasumsikan sebagai berikut:

- Anda mengunggah file teks 65 KB, dan konten file adalah string satu karakter yang terdiri dari huruf 'a'.
- Ukuran potongan adalah 64 KB. Akibatnya, payload diunggah dalam tiga potongan, 64 KB, 1 KB, dan potongan terakhir dengan 0 byte data potongan.
- Objek yang dihasilkan memiliki nama kuncichunk0object.txt.

- Anda meminta REDUCED_REDUNDANCY sebagai kelas penyimpanan dengan menambahkan header `x-amz-storage-class` permintaan.
- Transfer termasuk nilai checksum CRC32C sebagai header trailing.

Untuk informasi tentang tindakan API, lihat [PutObject](#). Sintaks permintaan umum adalah sebagai berikut:

```
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>
```

Langkah-langkah berikut menunjukkan perhitungan tanda tangan.

1. Tanda tangan benih - Buat String untuk Ditandatangani

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
```

Dalam permintaan kanonik, baris ketiga kosong karena tidak ada parameter kueri dalam permintaan. Baris terakhir adalah string konstan yang disediakan sebagai nilai Payload yang di-hash, yang harus sama dengan nilai. `x-amz-content-sha256` header

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
44d48b8c2f70eae815a0198cc73d7a546a73a93359c070abbaa5e6c7de112559
```

 Note

Untuk informasi tentang setiap baris dalam string yang akan ditandatangani, lihat diagram yang menjelaskan perhitungan tanda tangan benih.

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Tanda Tangan Benih

```
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
```

4. Header otorisasi

Header Otorisasi yang dihasilkan adalah sebagai berikut:

```
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. Potongan 1: (65536 byte, dengan nilai 97 untuk huruf 'a')

a. Potongan string untuk ditandatangani:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

Note

Untuk informasi tentang setiap baris dalam string yang akan ditandatangani, lihat diagram di topik sebelumnya ([Menghitung Tanda Tangan Benih](#)) yang menunjukkan berbagai komponen string untuk ditandatangani. Misalnya, tiga baris terakhir terdiri dari yang berikut:

- *previous-signature*
- `hash("")`
- `hash(current-chunk-data)`

b. Tanda tangan potongan:

```
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
```

c. Data potongan dikirim:

```
10000;chunk-
signature=b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
<65536-bytes>
```

6. Potongan 2: (1024 byte, dengan nilai 97 untuk huruf 'a')

a. Potongan string untuk ditandatangani:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
```

```
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. Tanda tangan potongan:

```
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
```

c. Data potongan dikirim:

```
400; chunk-
signature=1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
<1024-bytes>
```

7. Potongan 3: (0 byte data)

a. Potongan string untuk ditandatangani:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

b. Tanda tangan potongan:

```
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

c. Data potongan dikirim:

```
0; chunk-
signature=2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

8. Potongan 4: Header yang tertinggal

a. String potongan trailer untuk ditandatangani:

```
AWS4-HMAC-SHA256-TRAILER
20130524T000000Z
20130524/us-east-1/s3/aws4_request
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

```
1e376db7e1a34a8ef1c4bcee131a2d60a1cb62503747488624e10995f448d774
```

Note

Dua baris terakhir adalah `previous-signature` (tanda tangan potongan data 0 byte), hash (`trailing-checksum-header-name:base64-encoded-trailing-checksum-value\n`).

Hash dihitung sebagai berikut tanpa spasi:

- Nama header checksum di belakang
- Sebuah usus besar (:)
- Nilai checksum trailing yang dikodekan base64
- Karakter baris baru (\n).

Dalam contoh ini, di mana Anda menggunakan algoritma `crc32c` hash, dengan nilai checksum yang dikodekan base64 `s008/Q==`, Anda dapat mewakili perhitungan sebagai berikut: `hash( 'x-amz-checksum-crc32c:s008/Q==\n' )`

b. Tanda tangan potongan:

```
d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435
```

c. Data potongan dikirim:

```
x-amz-checksum-crc32c:s008/Q==  
x-amz-trailer-  
signature:d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435
```

Otentikasi Permintaan: Menggunakan Parameter Kueri (AWS Versi Tanda Tangan 4)

Seperti dijelaskan dalam ikhtisar otentikasi (lihat [Metode Autentikasi](#)), Anda dapat memberikan informasi otentikasi menggunakan parameter string kueri. Menggunakan parameter kueri untuk mengotentikasi permintaan berguna ketika Anda ingin mengungkapkan permintaan sepenuhnya dalam URL. Metode ini juga disebut sebagai penandatanganan URL.

Skenario kasus penggunaan untuk URL yang ditandatangani sebelumnya adalah Anda dapat memberikan akses sementara ke sumber daya Amazon S3 Anda. Misalnya, Anda dapat menyematkan URL yang telah ditandatangani sebelumnya di situs web Anda atau menggunakannya di klien baris perintah (seperti Curl) untuk mengunduh objek.

 Note

Anda juga dapat menggunakan AWS CLI untuk membuat URL yang ditandatangani sebelumnya. Untuk informasi selengkapnya, lihat [presign](#) dalam AWS CLI Referensi Perintah.

Berikut ini adalah contoh URL yang ditandatangani sebelumnya.

```
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>
```

Dalam contoh URL, perhatikan hal berikut:

- Umpan baris ditambahkan untuk keterbacaan.
- X-Amz-CredentialNilai dalam URL menunjukkan karakter “/” hanya untuk keterbacaan. Dalam praktiknya, itu harus dikodekan sebagai%2F. Contoh:

```
&X-Amz-Credential=<your-access-key-id>%2F20130721%2Fus-east-1%2Fs3%2Faws4_request
```

Tabel berikut menjelaskan parameter kueri di URL yang menyediakan informasi otentikasi.

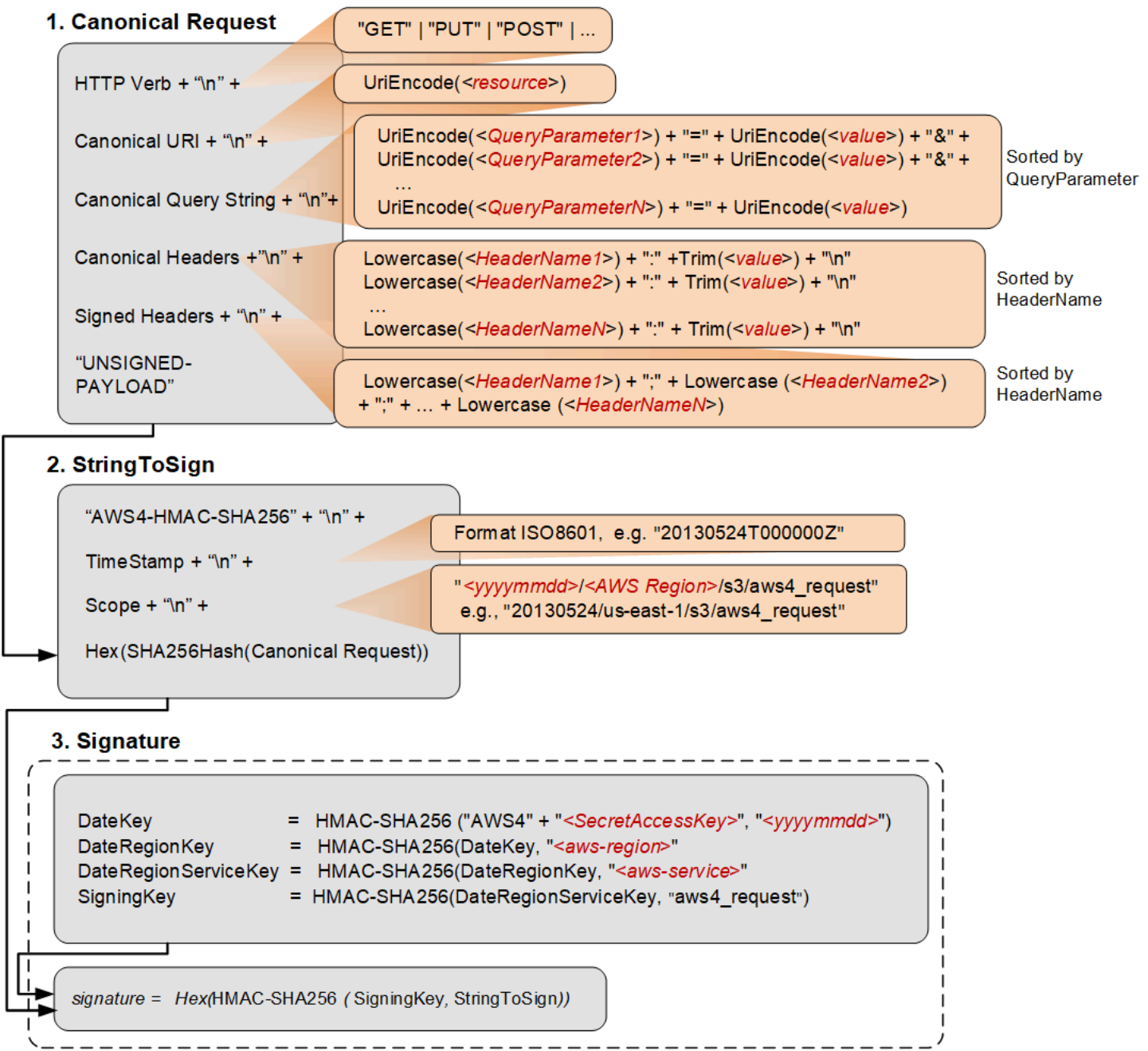
Nama Parameter String Kueri	Nilai Contoh
X-Amz-Algorithm	Mengidentifikasi versi T AWS anda Tangan dan algoritma yang Anda gunakan untuk menghitung tanda tangan. Untuk AWS Signature Version 4, Anda menetapkan nilai parameter ini ke <code>aws4-HMAC-SHA256</code> . String ini mengidentifikasi AWS Signature Version 4 (AWS 4) dan HMAC-SHA256 algoritma (HMAC-SHA256).
X-Amz-Credential	Selain ID kunci akses Anda, parameter ini juga menyediakan ruang lingkup (Wilayah AWS dan layanan) yang tandatanganannya valid. Nilai ini harus sesuai dengan cakupan yang Anda gunakan dalam perhitungan tanda tangan, dibahas di bagian berikut. Bentuk umum untuk nilai parameter ini adalah sebagai berikut: <pre><your-access-key-id> /<date>/<Wilayah AWS>/<AWS-service> /aws4_request</pre> Contoh: <pre>AKIAIOSFODNN7EXAMPLE/20130721/us-east-1/s3/aws4_request</pre> Untuk Amazon S3, <code>AWS-service</code> string adalah <code>s3</code>. Untuk daftar <code>AWS-region</code> string S3, lihat Wilayah dan Titik Akhir di Referensi Umum AWS.
X-Amz-Date	Format tanggal dan waktu harus mengikuti standar ISO 8601, dan harus diformat dengan format "<code>yyyyMMddT HHmmss Z</code>". Misalnya jika tanggal dan waktu adalah "08/01/2016 15:32:41.982-700" maka pertama-tama harus dikonversi ke UTC (Waktu Universal Terkoordinasi) dan kemudian dikirimkan sebagai "20160801T223241Z".

Nama Parameter String Kueri	Nilai Contoh
X-Amz-Expires	Menyediakan periode waktu, dalam detik, di mana URL yang telah ditandatangani sebelumnya valid. Misalnya, 86400 (24 jam). Nilai ini adalah bilangan bulat. Nilai minimum yang dapat Anda tetapkan adalah 1, dan maksimum adalah 604800 (tujuh hari). URL yang telah ditandatangani dapat berlaku maksimal tujuh hari karena kunci penandatanganan yang Anda gunakan dalam perhitungan tanda tangan berlaku hingga tujuh hari.
X-Amz-SignedHeaders	Daftar header yang Anda gunakan untuk menghitung tanda tangan. Header berikut diperlukan dalam perhitungan tanda tangan: • hostHeader HTTP. • Setiap x-amz-* header yang Anda rencanakan untuk ditambahkan ke permintaan.  Note Untuk keamanan tambahan, Anda harus menandatangani semua header permintaan yang Anda rencanakan untuk disertakan dalam permintaan Anda.
X-Amz-Signature	Menyediakan tanda tangan untuk mengotentikasi permintaan Anda. Tanda tangan ini harus sesuai dengan tanda tangan yang dihitung Amazon S3; jika tidak, Amazon S3 menolak permintaan tersebut. Sebagai contoh, 733255ef022bec3f2a8701cd61d4b371f3f28c9f193a1f02279211d48d5193d7 . Perhitungan tanda tangan dijelaskan di bagian berikut.

Nama Parameter String Kueri	Nilai Contoh
X-Amz-Security-Token	Parameter kredensi opsional jika menggunakan kredensial yang bersumber dari layanan STS.

Menghitung Tanda Tangan

Diagram berikut menggambarkan proses perhitungan tanda tangan.



Tabel berikut menjelaskan fungsi yang ditunjukkan dalam diagram. Anda perlu menerapkan kode untuk fungsi-fungsi ini.

Fungsi	Deskripsi
Lowercase()	Ubah string menjadi huruf kecil.
Hex()	Pengkodean huruf kecil 16.

Fungsi	Deskripsi
SHA256Hash()	Fungsi hash kriptografi Secure Hash Algorithm (SHA).
HMAC-SHA256()	Menghitung HMAC dengan menggunakan algoritma SHA256 dengan kunci penandatanganan yang disediakan. Ini adalah tanda tangan terakhir.
Trim()	Hapus spasi kosong di depan atau di belakang.

Fungsi	Deskripsi
<code>UriEncode()</code>	URI mengkodekan setiap byte. <code>UriEncode()</code> harus menegakkan aturan berikut: • URI mengkodekan setiap byte kecuali karakter tanpa syarat: 'A'-'Z', 'a'-'z', '0'-'9', '-', '.', '_', dan '~'. • Karakter spasi adalah karakter yang dicadangkan dan harus dikodekan sebagai "%20" (dan bukan sebagai "+"). • Setiap byte yang dikodekan URI dibentuk oleh '%' dan nilai heksadesimal dua digit byte. • Huruf dalam nilai heksadesimal harus huruf besar, misalnya "%1A". • Enkode karakter garis miring ke depan, '/', di mana-mana kecuali dalam nama kunci objek. Misalnya, jika nama kunci objek adalah <code>photos/Jan/sample.jpg</code>, garis miring ke depan dalam nama kunci tidak dikodekan.  Important <code>UriEncode</code> Fungsi standar yang disediakan oleh platform pengembangan Anda mungkin tidak berfungsi karena perbedaan dalam implementasi dan ambiguitas terkait dalam RFC yang mendasarinya. Kami menyarankan Anda menulis <code>UriEncode</code> fungsi kustom Anda sendiri untuk memastikan bahwa pengkodean Anda akan berfungsi. Berikut ini adalah contoh fungsi <code>UriEncode()</code> di 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>

Fungsi	Deskripsi
	<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>

Untuk informasi selengkapnya tentang proses penandatanganan (detail pembuatan permintaan kanonik, string untuk ditandatangani, dan perhitungan tanda tangan), lihat [Perhitungan Tanda Tangan untuk Header Otorisasi: Mentransfer Payload dalam Satu Potongan \(AWS Versi Tanda Tangan 4\)](#). Prosesnya umumnya sama kecuali bahwa pembuatan CanonicalRequest dalam URL yang telah ditandatangani berbeda sebagai berikut:

- Anda tidak menyertakan hash payload di Canonical Request, karena ketika Anda membuat URL yang telah ditandatangani sebelumnya, Anda tidak mengetahui konten payload karena URL digunakan untuk mengunggah payload sewenang-wenang. Sebagai gantinya, Anda menggunakan string konstan UNSIGNED-PAYLOAD.
- Canonical Query String harus menyertakan semua parameter kueri dari tabel sebelumnya kecuali untuk X-Amz-Signature.
- Untuk S3, Anda harus menyertakan parameter ku X-Amz-Security-Token eri di URL jika menggunakan kredenSIAL yang bersumber dari layanan STS.
- Header kanonik harus menyertakan header HTTPHost. Jika Anda berencana untuk menyertakan salah satu x-amz-* header, header ini juga harus ditambahkan untuk perhitungan tanda tangan. Anda dapat secara opsional menambahkan semua header lain yang Anda rencanakan untuk disertakan dalam permintaan Anda. Untuk keamanan tambahan, Anda harus menandatangani header sebanyak mungkin. Jika Anda menambahkan header yang ditandatangani yang juga merupakan parameter kueri yang ditandatangani, dan nilainya berbeda, Anda akan menerima InvalidRequest kesalahan karena input bertentangan.

Sebuah contoh

Misalkan Anda memiliki objek `test.txt` di `examplebucket` ember Anda. Anda ingin berbagi objek ini dengan orang lain untuk jangka waktu 24 jam (86400 detik) dengan membuat URL yang telah ditandatangani sebelumnya.

```
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>
```

Langkah-langkah berikut menggambarkan terlebih dahulu perhitungan tanda tangan dan kemudian konstruksi URL yang telah ditandatangani sebelumnya. Contoh membuat asumsi tambahan berikut:

- Stempel waktu permintaan adalah. `Fri, 24 May 2013 00:00:00 GMT`
- Bucket berada di wilayah AS Timur (Virginia Utara), dan kreden Scope si serta Signing Key perhitungan digunakan `us-east-1` sebagai penentu wilayah. Untuk informasi selengkapnya, lihat [Wilayah dan Titik Ak](#) hir di Referensi Umum AWS.

Anda dapat menggunakan contoh ini sebagai kasus uji untuk memverifikasi tanda tangan yang dihitung kode Anda; namun, Anda harus menggunakan nama bucket yang sama, kunci objek, stempel waktu, dan kredenSIAL contoh berikut:

Parameter	Nilai
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJa1rXUtnFEMI/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
3bfa292879f6447bbcda7001decf97f4a54dc650c8942174ae0a9121cf58ad04
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Tanda tangan

```
aeed9bbccd4d02ee5c0109b86d86835f995330da4c265957d157751f604d404
```

Sekarang Anda memiliki semua informasi untuk membangun URL yang telah ditandatangani sebelumnya. URL yang dihasilkan untuk contoh ini ditampilkan sebagai berikut (Anda dapat menggunakan ini untuk membandingkan URL yang telah ditandatangani sebelumnya):

```
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=aeed9bbccd4d02ee5c0109b86d86835f995330da4c265957d157751f604d404
```


- Objek PUT — Contoh terpisah menggambarkan mengunggah payload penuh sekaligus dan mengunggah payload dalam potongan. Untuk informasi tentang menggunakan header Otorisasi untuk otentikasi, lihat [Permintaan Otentikasi: Menggunakan Header Otorisasi \(AWS Versi Tanda Tangan 4\)](#).
- Objek GET — Contoh ini menghasilkan URL yang ditandatangani sebelumnya untuk mendapatkan objek. Parameter kueri memberikan tanda tangan dan informasi otentikasi lainnya. Pengguna dapat menempelkan URL yang telah ditandatangani sebelumnya di browser mereka untuk mengambil objek, atau Anda dapat menggunakan URL untuk membuat tautan yang dapat diklik. Untuk informasi tentang menggunakan parameter kueri untuk otentikasi, lihat [Otentikasi Permintaan: Menggunakan Parameter Kueri \(AWS Versi Tanda Tangan 4\)](#).

Sisa bagian ini menjelaskan contoh dalam Java dan C#. Topik termasuk instruksi untuk mengunduh sampel dan untuk mengeksekusinya.

Contoh Perhitungan Tanda Tangan Menggunakan Java (AWS Versi Tanda Tangan 4)

Sampel Java yang menunjukkan perhitungan tanda tangan dapat diunduh di <https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AWSS3SigV4JavaSamples.zip>. Perhatikan bahwa sampel ini menggunakan AWS SDK untuk Java v1, dan kami sarankan menggunakan [AWS SDK untuk Java v2](#) untuk aplikasi baru. Di `RunAllSamples.java`, `main()` fungsi mengeksekusi permintaan sampel untuk membuat objek, mengambil objek, dan membuat URL yang ditandatangani sebelumnya untuk objek tersebut. Sampel membuat objek dari string teks yang disediakan dalam kode:

```
PutS3ObjectSample.putS3Object(bucketName, regionName, awsAccessKey, awsSecretKey);
GetS3ObjectSample.getS3Object(bucketName, regionName, awsAccessKey, awsSecretKey);
PresignedUrlSample.getPresignedUrlToS3Object(bucketName, regionName, awsAccessKey,
awsSecretKey);
PutS3ObjectChunkedSample.putS3ObjectChunked(bucketName, regionName, awsAccessKey,
awsSecretKey);
```

Untuk menguji contoh di Linux-based komputer

Instruksi berikut adalah untuk sistem operasi Linux.

1. Di terminal, arahkan ke direktori yang berisi `AWSS3SigV4JavaSamples.zip`.
2. Ekstrak file.zip.

3. Di editor teks, buka file `./com/amazonaws/services/s3/samples/RunAllSamples.java`. Perbarui kode dengan informasi berikut:

- Nama bucket tempat objek baru dapat dibuat.

 Note

Contoh-contoh menggunakan permintaan gaya yang dihosting virtual untuk mengakses bucket. Untuk menghindari potensi kesalahan, pastikan nama bucket Anda sesuai dengan aturan penamaan bucket seperti yang dijelaskan dalam Pembatasan dan Batasan Bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

- Wilayah AWS di mana ember itu berada.

Jika bucket berada di wilayah AS Timur (Virginia Utara), gunakan `us-east-1` untuk menentukan wilayah. Untuk daftar lainnya Wilayah AWS, buka [Amazon Simple Storage Service \(S3\)](#) di Referensi Umum AWS.

4. Mengkompilasi kode sumber dan menyimpan kelas yang dikompilasi ke dalam `bin/` direktori.

```
javac -d bin -source 6 -verbose com
```

5. Ubah direktori ke `bin/`, lalu jalankan `RunAllSamples`.

```
java com.amazonaws.services.s3.sample.RunAllSamples
```

Kode menjalankan semua metode di `main()`. Untuk setiap permintaan, output akan menampilkan permintaan kanonik, string yang akan ditandatangani, dan tanda tangan.

Contoh Perhitungan Tanda Tangan Menggunakan C # (AWS Versi Tanda Tangan 4)

Sampel C# yang menunjukkan perhitungan tanda tangan dapat diunduh di https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AmazonS3SigV4_Samples_CSharp.zip.

Perhatikan bahwa sampel ini menggunakan AWS SDK untuk .NET v1, dan sebaiknya gunakan [AWS SDK untuk .NET v3](#) untuk aplikasi baru. Di `Program.cs`, `main()` fungsi mengeksekusi

permintaan sampel untuk membuat objek, mengambil objek, dan membuat URL yang ditandatangani sebelumnya untuk objek tersebut. Kode untuk perhitungan tanda tangan ada di \Signers folder.

```
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");
```

Untuk menguji contoh dengan Microsoft Visual Studio 2010 atau yang lebih baru

1. Ekstrak file.zip.
2. Mulai Visual Studio, lalu buka file.sln.
3. Perbarui App.config file dengan kredentif keamanan yang valid.
4. Perbarui kode sebagai berikut:
 - DiProgram.cs, berikan nama bucket dan Wilayah AWS tempat bucket berada. Sampel membuat objek di bucket ini.
5. Jalankan kode tersebut.
6. Untuk memverifikasi bahwa objek telah dibuat, salin URL yang telah ditandatangani sebelumnya yang dibuat program, lalu tempelkan di jendela browser.

Permintaan Otentikasi: Ung Browser-Based gah Menggunakan POST (AWS Versi Tanda Tangan 4)

Amazon S3 mendukung permintaan HTTP POST sehingga pengguna dapat mengunggah konten langsung ke Amazon S3. Menggunakan HTTP POST untuk mengunggah konten menyederhanakan unggahan dan mengurangi latensi unggahan di mana pengguna mengunggah data untuk disimpan di Amazon S3. Bagian ini menjelaskan cara Anda mengotentikasi permintaan HTTP POST. Untuk informasi selengkapnya tentang permintaan HTTP POST, cara membuat formulir, membuat kebijakan POST, dan contoh, lihat [Browser-Based Unggah Menggunakan POST \(AWS Versi Tanda Tangan 4\)](#).

Untuk mengotentikasi permintaan HTTP POST, lakukan hal berikut:

1. Formulir harus menyertakan bidang berikut untuk memberikan tanda tangan dan informasi relevan yang dapat digunakan Amazon S3 untuk menghitung ulang tanda tangan setelah menerima permintaan:

Nama Elemen	Deskripsi
<code>policy</code>	Kebijakan Base64-encoded keamanan yang menjelaskan apa yang diizinkan dalam permintaan. Untuk perhitungan tanda tangan, kebijakan ini adalah string yang Anda tandatangani. Amazon S3 harus mendapatkan kebijakan ini sehingga dapat menghitung ulang tanda tangan.
<code>x-amz-algorithm</code>	Algoritma penandatanganan yang digunakan. Untuk AWS Signature Version 4, nilainya adalah <code>AWS4-HMAC-SHA256</code> .
<code>x-amz-credential</code>	Selain ID kunci akses Anda, ini menyediakan informasi cakupan yang Anda gunakan dalam menghitung kunci penandatanganan untuk perhitungan tanda tangan. Ini adalah string dari bentuk berikut: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Contoh: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . .</pre> Untuk Amazon S3, string <code>aws-service</code> adalah <code>s3</code>. Untuk daftar <code>aws-region</code> string Amazon S3, lihat Wil ayah dan Titik Akhir di Referensi Umum AWS.

Nama Elemen	Deskripsi
x-amz-date	Ini adalah nilai tanggal dalam format ISO8601. Misalnya, 20130728T000000Z . Ini adalah tanggal yang sama yang Anda gunakan dalam membuat kunci penandatanganan. Nilai ini juga harus sama dengan yang Anda berikan dalam kebijakan (x-amz-date) yang Anda tandatangani.
x-amz-signature	(AWS Signature Version 4) HMAC-SHA256 Hash dari kebijakan keamanan. Untuk informasi selengkapnya tentang opsi untuk tanda tangan, lihat Men ambahkan tanda tangan ke permintaan HTTP di Referensi Umum AWS.

2. Kebijakan POST harus mencakup elemen-elemen berikut:

Nama Elemen	Deskripsi
x-amz-algorithm	Algoritma penandatanganan yang Anda gunakan untuk menghitung tanda tangan. Untuk AWS Signature Version 4, nilainya adalah AWS4-HMAC-SHA256 .
x-amz-credential	Selain ID kunci akses Anda, ini menyediakan informasi cakupan yang Anda gunakan dalam menghitung kunci penandatanganan untuk perhitungan tanda tangan. Ini adalah string dari bentuk berikut: <i><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</i> Misalnya,

Nama Elemen	Deskripsi
	AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . .
x-amz-date	Nilai tanggal yang ditentukan dalam string berformat ISO8601. Misalnya, "20130728T000000Z". Tanggal harus sama dengan yang Anda gunakan saat membuat kunci penandatanganan untuk perhitungan tanda tangan.

3. Untuk perhitungan tanda tangan, kebijakan POST adalah string yang akan ditandatangani.

Menghitung Tanda Tangan

Diagram berikut menggambarkan proses perhitungan tanda tangan.



Untuk Menghitung Tanda Tangan

1. Buat kebijakan menggunakan UTF-8 pengkodean.
2. Ubah UTF-8-encoded kebijakan menjadi Base64. Hasilnya adalah string yang akan ditandatangani.
3. Buat tanda tangan sebagai HMAC-SHA256 hash dari string yang akan ditandatangani. Anda akan memberikan kunci penandatanganan sebagai kunci untuk fungsi hash.
4. Enkode tanda tangan dengan menggunakan pengkodean hex.

Untuk informasi selengkapnya tentang membuat formulir HTML, kebijakan keamanan, dan contoh, lihat subtopik berikut:

- [Membuat Formulir HTML \(Menggunakan AWS Versi Tanda Tangan 4\)](#)
- [Kebijakan POST](#)
- [Contoh: Browser-Based Unggah menggunakan HTTP POST \(Menggunakan AWS Versi Tanda Tangan 4\)](#)

Kunci Kebijakan Khusus Otentikasi Tanda Tangan Amazon S3 Versi 4

Tabel berikut menunjukkan kunci kebijakan terkait otentikasi Amazon S3 Signature Versi 4 yang dapat ada di kebijakan Amazon S3. Dalam kebijakan bucket, Anda dapat menambahkan kondisi ini untuk menerapkan perilaku tertentu saat permintaan diautentikasi menggunakan Signature Versi 4. Untuk kebijakan-kebijakan contoh, lihat [Contoh kebijakan bucket menggunakan kunci kondisi terkait Signature Versi 4](#).

Kunci yang Berlaku	Deskripsi
s3:signatureversion	Mengidentifikasi versi Tanda AWS Tangan yang ingin Anda dukung untuk permintaan yang diautentikasi. Untuk permintaan yang diautentikasi, Amazon S3 mendukung Tanda Tangan Versi 4 dan Tanda Tangan Versi 2. Anda dapat menambahkan kondisi ini dalam kebijakan

Kunci yang Berlaku	Deskripsi
	bucket untuk memerlukan versi tanda tangan tertentu. Nilai valid: "AWS" mengidentifikasi Tanda Tangan Versi 2 "AWS4-HMAC-SHA256" mengidentifikasi Tanda Tangan Versi 4
s3:authType	Amazon S3 mendukung berbagai metode otentikasi (lihat Permintaan Otentikasi (AWS Versi Tanda Tangan 4)). Anda dapat menggunakan kunci kondisi ini secara opsional untuk membatasi permintaan masuk untuk menggunakan metode otentikasi tertentu. Misalnya, Anda hanya dapat mengizinkan <code>Authorization</code> header HTTP untuk digunakan dalam otentikasi permintaan. Nilai valid: REST-HEADER REST-QUERY-STRING POST

Kunci yang Berlaku	Deskripsi
s3:signatureAge	Lamanya waktu, dalam milidetik, tanda tangan valid dalam permintaan autentikasi. Kondisi ini berfungsi untuk: • URL yang ditandatangani sebelumnya — di mana kondisi paling ketat menang. Untuk informasi selengkapnya, lihat Bek erja dengan URL yang ditandatangani sebelumnya. • POST yang telah ditandatangani sebelumnya — unggah file langsung ke S3 menggunakan POST yang telah ditandatangani sebelumnya. Untuk informasi selengkapnya, lihat Kebijakan POST Amazon S3. Di Signature Version 2, nilai ini selalu disetel ke 0. Di Signature Versi 4, kunci penandatanganan berlaku hingga tujuh hari. Karena itu, tanda tangan juga berlaku hingga tujuh hari. Anda dapat menggunakan syarat ini untuk lebih lanjut membatasi umur tanda tangan. Untuk informasi selengkapnya, lihat Pengantar Permintaan Penandatanganan. Nilai contoh: 100

Kunci yang Berlaku	Deskripsi
s3:x-amz-content-sha256	Anda dapat menggunakan kunci syarat ini untuk melarang konten yang tidak ditandatangani di bucket Anda. Ketika Anda menggunakan Signature Version 4 untuk permintaan yang menggunakan Authorization header Authorization, Anda menambahkan x-amz-content-sha256 header x-amz-content-sha256 dalam perhitungan tanda tangan, lalu kemudian menetapkan nilainya ke muatan hash. Perhatikan bahwa kunci kondisi ini tidak mendukung x-amz-content-sha256 header sebagai parameter string kueri. Anda dapat menggunakan kunci kondisi ini dalam kebijakan bucket Anda untuk menolak unggahan apa pun yang tidak ditandatangani payload. Misalnya, Anda dapat menolak unggahan yang menggunakan Authorization header untuk mengotentikasi permintaan tetapi tidak menandatangani payload. Untuk informasi selengkapnya, lihat Perhitungan Tanda Tangan untuk Header Otorisasi: Mentransfer Payload dalam Satu Potongan (AWS Versi Tanda Tangan 4). Nilai valid: UNSIGNED-PAYLOAD

Contoh kebijakan bucket menggunakan kunci kondisi terkait Signature Versi 4

Kebijakan bucket berikut menolak permintaan URL Amazon S3 yang telah ditandatangani sebelumnya pada objek `examplebucket` jika tanda tangan berusia lebih dari sepuluh menit.

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
        }
      }
    }
  ]
}
```

Kebijakan bucket berikut hanya mengizinkan permintaan yang menggunakan Authorization header untuk otentikasi permintaan. Setiap POST atau permintaan URL yang telah ditandatangani sebelumnya akan ditolak.

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"
        }
      }
    }
  ]
}
```

```

    }
  }
]
}

```

Kebijakan bucket berikut menolak permintaan yang menggunakan URL yang telah ditandatangani sebelumnya untuk otentikasi permintaan:

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"
        }
      }
    }
  ]
}

```

Kebijakan bucket berikut menolak upload dengan payload yang tidak ditandatangani:

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 Unggah Menggunakan POST (AWS Versi Tanda Tangan 4)

Bagian ini membahas cara mengunggah file langsung ke Amazon S3 melalui browser menggunakan permintaan HTTP POST. Ini juga berisi informasi tentang cara menggunakan JavaScript perpustakaan AWS Amplify untuk unggahan file berbasis browser ke Amazon S3.

Topik

- [Objek POST](#)
- [Pemulihan objek POST](#)
- [Browser-Based Unggah Menggunakan HTTP POST](#)
- [Menghitung Tanda Tangan](#)
- [Membuat Formulir HTML \(Menggunakan AWS Versi Tanda Tangan 4\)](#)
- [Kebijakan POST](#)
- [Contoh: Browser-Based Unggah menggunakan HTTP POST \(Menggunakan AWS Versi Tanda Tangan 4\)](#)
- [Browser-based mengunggah ke Amazon S3 menggunakan AWS Memperkuat perpustakaan](#)

Objek POST

Deskripsi

POST Operasi menambahkan objek ke bucket tertentu dengan menggunakan formulir HTML. POST adalah bentuk alternatif PUT yang memungkinkan unggahan berbasis browser sebagai cara memasukkan objek ke dalam bucket. Parameter yang diteruskan PUT melalui header HTTP malah diteruskan sebagai bidang formulir ke POST dalam badan pesan yang dikodekan multipart/form-data. Untuk menambahkan objek ke bucket, Anda harus memiliki WRITE akses pada bucket. Amazon S3 tidak pernah menyimpan sebagian objek. Jika Anda menerima respons yang berhasil, Anda dapat yakin bahwa seluruh objek disimpan.

Amazon S3 adalah sistem terdistribusi. Kecuali Anda telah mengaktifkan pembuatan versi untuk bucket, jika Amazon S3 menerima beberapa permintaan penulisan untuk objek yang sama secara bersamaan, hanya versi terakhir dari objek yang ditulis yang disimpan.

Untuk memastikan bahwa data tidak rusak saat melintasi jaringan, gunakan kolom Content-MD5 formulir. Saat Anda menggunakan bidang formulir ini, Amazon S3 memeriksa objek terhadap nilai MD5 yang disediakan. Jika tidak cocok, Amazon S3 mengembalikan kesalahan. Selain itu, Anda dapat menghitung nilai MD5 saat memposting objek ke Amazon S3 dan membandingkan nilai yang dikembalikan ETag ke nilai MD5 yang dihitung. ETag hanya mencerminkan perubahan pada isi objek, bukan metadatanya.

Note

Untuk mengonfigurasi aplikasi Anda untuk mengirim header permintaan sebelum mengirim isi permintaan, gunakan kode status HTTP 100 (Lanjutkan). Untuk POST operasi, menggunakan kode status ini membantu Anda menghindari pengiriman isi pesan jika pesan ditolak berdasarkan header (misalnya, karena kegagalan otentikasi atau pengalihan). Untuk informasi selengkapnya tentang kode status HTTP 100 (Lanjutkan), buka Bagian 8.2.3 dari <http://www.ietf.org/rfc/rfc2616.txt>.

Amazon S3 secara otomatis mengenkripsi semua objek baru yang diunggah ke bucket S3. Pengaturan enkripsi objek yang diunggah tergantung pada konfigurasi enkripsi default dari bucket tujuan. Secara default, semua bucket memiliki konfigurasi enkripsi default yang menggunakan enkripsi sisi server dengan kunci terkelola Amazon S3 (SSE-S3).

Jika bucket tujuan memiliki konfigurasi enkripsi yang menggunakan enkripsi sisi server dengan kunci AWS Key Management Service (AWS KMS), enkripsi sisi server lapis ganda dengan kunci (SSE-KMS), atau AWS KMS kunci enkripsi yang disediakan pelanggan (DSSE-KMS), Amazon S3 menggunakan kunci KMS yang sesuai atau kunci yang disediakan pelanggan untuk mengenkripsi objek yang diunggah. SSE-C Saat mengunggah objek, jika Anda ingin mengubah pengaturan enkripsi objek yang diunggah, Anda dapat menentukan jenis enkripsi sisi server. Anda dapat mengkonfigurasi SSE-S3, SSE-KMS, DSSE-KMS, atau SSE-C. Untuk informasi selengkapnya, lihat [Melindungi data menggunakan enkripsi sisi server](#) di Panduan Pengguna Amazon Simple Storage Service.

Important

Saat membuat permintaan Anda, pastikan bahwa file bidang tersebut adalah bidang terakhir dalam formulir.

Penentuan versi

Jika Anda mengaktifkan pembuatan versi untuk bucket, POST secara otomatis menghasilkan ID versi unik untuk objek yang ditambahkan. Amazon S3 mengembalikan ID ini dalam respons dengan menggunakan header `x-amz-version-id` respons.

Jika Anda menangguhkan pembuatan versi untuk bucket, Amazon S3 selalu menggunakan ID versi objek yang disimpan dalam bucket `null` sebagai ID versi.

Untuk informasi selengkapnya tentang mengembalikan status versi bucket, lihat [GetBucketVersioning](#).

Amazon S3 adalah sistem terdistribusi. Jika Anda mengaktifkan pembuatan versi untuk bucket dan Amazon S3 menerima beberapa permintaan penulisan untuk objek yang sama secara bersamaan, semua versi objek disimpan.

Untuk melihat permintaan sampel yang menggunakan pembuatan versi, lihat [Permintaan Sam](#) [pel](#).

Permohonan

Sintaksis

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--
```

Parameter Permintaan

Implementasi operasi ini tidak menggunakan parameter permintaan.

Bidang Formulir

Operasi ini dapat menggunakan kolom formulir berikut.

Nama	Deskripsi	Diperlukan
AWSSecretAccessKey	ID kunci AWS akses pemilik bucket yang memberikan akses pengguna Anonim untuk permintaan yang memenuhi kumpulan batasan dalam kebijakan. Tipe: String Default: Tidak Ada Kendala: Diperlukan jika dokumen kebijakan disertakan dengan permintaan.	Bersyarat
acl	Daftar kontrol akses Amazon S3 (ACL) yang ditentukan. Jika ACL yang ditentukan tidak valid, kesalahan dihasilkan. Untuk informasi selengkapnya tentang ACL, lihat ikhtisar daftar kontrol akses	Tidak

Nama	Deskripsi	Diperlukan
	(ACL) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Tipe: String Default: privat Nilai Valid: 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	He REST-specific ader. Untuk informasi selengkapnya, lihat PutObject. Tipe: String Default: Tidak Ada	Tidak
file	File atau konten teks. File atau konten teks harus menjadi bidang terakhir dalam formulir. Anda tidak dapat mengunggah lebih dari satu file sekaligus. Jenis: File atau konten teks Default: Tidak Ada	Ya

Nama	Deskripsi	Diperlukan
key	Nama kunci yang diunggah. Untuk menggunakan nama file yang disediakan oleh pengguna, gunakan <code>\${filename}</code> variabel. Misalnya, jika pengguna bernama Mary mengunggah file <code>example.jpg</code> dan Anda tentukan <code>/user/mary/\${filename}</code>, nama kuncinya adalah <code>user/mary/example.jpg</code>. Untuk informasi selengkapnya, lihat Kunci objek dan metadata di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Tipe: String Default: Tidak Ada	Ya
policy	Kebijakan keamanan yang menjelaskan apa yang diizinkan dalam permintaan. Permintaan tanpa kebijakan keamanan dianggap anonim dan hanya berfungsi pada bucket yang dapat ditulis secara publik. Untuk informasi selengkapnya, lihat formulir HTML dan Unggah contoh di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Tipe: String Default: Tidak Ada Kendala: Kebijakan keamanan diperlukan jika bucket tidak dapat ditulis secara publik.	Bersyarat

Nama	Deskripsi	Diperlukan
<code>success_action_redirect</code> , <code>redirect</code>	URL tempat klien dialihkan setelah unggahan berhasil. Jika <code>success_action_redirect</code> tidak ditentukan, maka Amazon S3 akan mengembalikan jenis dokumen kosong yang ditentukan dalam bidang <code>success_action_status</code>. Jika Amazon S3 tidak dapat menafsirkan URL, ia bertindak seolah-olah bidang tidak ada. Jika pengunggahan gagal, maka Amazon S3 akan menampilkan pesan kesalahan dan tidak akan mengalihkan pengguna ke URL. Tipe: String Default: Tidak Ada  Note Nama <code>redirect</code> bidang tidak digunakan lagi, dan dukungan untuk nama <code>redirect</code> bidang akan dihapus di masa mendatang.	Tidak

Nama	Deskripsi	Diperlukan
success_action_status	Jika Anda tidak menentukan <code>success_action_redirect</code> , kode status dikembalikan ke klien saat upload berhasil. Bidang ini menerima nilai <code>200</code> atau <code>201</code>, atau <code>204</code> (default). Jika nilainya disetel ke <code>200</code> atau <code>204</code>, Amazon S3 mengembalikan dokumen kosong dengan kode status <code>200</code> atau <code>204</code>. Jika nilainya disetel ke <code>201</code>, Amazon S3 mengembalikan dokumen XML dengan kode status <code>201</code>. Jika nilai tidak disetel atau jika disetel ke nilai yang tidak valid, Amazon S3 mengembalikan dokumen kosong dengan kode status <code>204</code>. Tipe: String Default: Tidak Ada	Tidak

Nama	Deskripsi	Diperlukan
tagging	Kumpulan tag yang ditentukan untuk ditambahkan ke objek. Untuk menambahkan tag, gunakan skema pengkodean berikut. <pre><Tagging> <TagSet> <Tag> <Key>TagName</Key> <Value>TagValue</Value> </Tag> ... </TagSet> </Tagging></pre>	Tidak
Untuk informasi selengkapnya, lihat Penandaan objek di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Tipe: String Default: Tidak Ada		

Nama	Deskripsi	Diperlukan
<code>x-amz-storage-class</code>	Kelas penyimpanan yang digunakan untuk menyimpan objek. Jika Anda tidak menentukan kelas, Amazon S3 menggunakan kelas penyimpanan default, STANDARD. Amazon S3 mendukung kelas penyimpanan lainnya. Untuk informasi selengkapnya, lihat Kelas penyimpanan di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Tipe: String Default: STANDARD Nilai yang valid: REDUCED_REDUNDANCY EXPRESS_ONEZONE DEEP_ARCHIVE GLACIER GLACIER_IR INTELLIGENT_TIERING ONEZONE_IA STANDARD STANDARD_IA	Tidak
<code>x-amz-meta-*</code>	Header yang dimulai dengan awalan ini adalah metadata yang ditentukan pengguna. Masing-masing disimpan dan dikembalikan sebagai satu set pasangan kunci-nilai. Amazon S3 tidak memvalidasi atau menafsirkan metadata yang ditentukan pengguna. Untuk informasi selengkapnya, lihat PutObject. Tipe: String Default: Tidak Ada	Tidak

Nama	Deskripsi	Diperlukan
x-amz-security-token	Token DevPay keamanan Amazon. Setiap permintaan yang menggunakan Amazon DevPay memerlukan dua bidang x-amz-security-token formulir: satu untuk token produk dan satu untuk token pengguna. Tipe: String Default: Tidak Ada	Tidak
x-amz-signature	(AWS Signature Version 4) HMAC-SHA256 Hash dari kebijakan keamanan. Tipe: String Default: Tidak Ada	Bersyarat

Nama	Deskripsi	Diperlukan
x-amz-website-redirect-location	Jika bucket dikonfigurasi sebagai situs web, bidang ini mengalihkan permintaan untuk objek ini ke objek lain dalam bucket yang sama atau ke URL eksternal. Amazon S3 menyimpan nilai header ini dalam metadata objek. Untuk informasi tentang metadata objek, lihat Kunci objek dan metadata di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Dalam contoh berikut, header permintaan menetapkan pengalihan ke objek (anotherPage.html) dalam bucket yang sama: <pre>x-amz-website-redirect-location: /anotherPage.html</pre> Dalam contoh berikut, header permintaan mengatur pengalihan objek ke situs web lain: <pre>x-amz-website-redirect-location: http://www.example.com/</pre> Untuk informasi selengkapnya tentang hosting situs web di Amazon S3, lihat Hosting situs web di Amazon S3 dan Cara mengonfigurasi pengalihan halaman situs web di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Tipe: String Default: Tidak Ada Kendala: Nilai harus diawali dengan/,http://, atauhttps://. Panjang nilai dibatasi hingga 2 KB.	Tidak

Bidang Formulir Permintaan Checksum Tambahan

Saat mengunggah objek, Anda dapat menentukan berbagai checksum yang ingin Anda gunakan untuk memverifikasi integritas data Anda. Anda dapat menentukan satu algoritma checksum tambahan untuk Amazon S3 untuk digunakan. Untuk informasi selengkapnya tentang nilai checksum tambahan, lihat Memer [iksa integritas objek](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Nama	Deskripsi	Diperlukan
x-amz-checksum-algorithm	Menunjukkan algoritma yang digunakan untuk membuat checksum untuk objek. Jika nilai ditentukan, Anda harus menyertakan header checksum yang cocok. Jika tidak, permintaan Anda akan menghasilkan kesalahan 400. Nilai yang mungkin termasuk CRC32, CRC32C, SHA1, dan SHA256.	Tidak
x-amz-checksum-crc32	Menentukan checksum CRC32 32-bit yang dikodekan base64 dari objek. Parameter ini diperlukan jika nilainya x-amz-checksum-algorithm adalahCRC32.	Bersyarat
x-amz-checksum-crc32c	Menentukan checksum objek CRC32C 32-bit yang dikodekan base64. Parameter ini diperlukan jika nilainya x-amz-checksum-algorithm adalahCRC32C.	Bersyarat
x-amz-checksum-sha1	Menentukan ringkasan 160-bit yang dikodekan base64 dari objek SHA-1 . Parameter ini diperlukan jika nilainya x-amz-checksum-algorithm adalahSHA1.	Bersyarat
x-amz-checksum-sha256	Menentukan intisari objek 256-bit yang dikodekan base64 SHA-256 .	Bersyarat

Nama	Deskripsi	Diperlukan

Parameter ini diperlukan jika nilainya `x-amz-checksum-algorithm` adalah `SHA256`.

Server-Side Bidang Formulir Permintaan Khusus Enkripsi

Server-side Enkripsi adalah enkripsi data saat diam. Amazon S3 mengenkripsi data Anda saat menulisnya ke disk di pusat AWS data dan mendekripsi data Anda saat Anda mengaksesnya. Saat mengunggah objek, Anda dapat menentukan jenis enkripsi sisi server yang ingin digunakan Amazon S3 untuk mengenkripsi objek.

Ada empat jenis enkripsi sisi server:

- Server-side enkripsi dengan kunci terkelola Amazon S3 (SSE-S3) — Mulai Mei 2022, semua bucket Amazon S3 memiliki enkripsi yang dikonfigurasi secara default. Opsi default untuk enkripsi sisi server adalah dengan. SSE-S3 Setiap objek dienkripsi dengan kunci unik. Sebagai perlindungan tambahan, SSE-S3 mengenkripsi kunci itu sendiri dengan kunci root yang secara teratur diputar. SSE-S3 menggunakan salah satu sandi blok terkuat yang tersedia, 256-bit Advanced Encryption Standard (AES-256), untuk mengenkripsi data Anda.
- Server-side enkripsi dengan AWS KMS kunci (SSE-KMS) — SSE-KMS disediakan melalui integrasi AWS KMS layanan dengan Amazon S3. Dengan AWS KMS, Anda memiliki kontrol lebih besar atas kunci Anda. Misalnya, Anda dapat melihat kunci terpisah, mengedit kebijakan kontrol, dan mengikuti kunci di AWS CloudTrail. Selain itu, Anda dapat membuat dan mengelola CMK atau menggunakan Kunci yang dikelola AWS yang unik bagi Anda, layanan Anda, dan Wilayah Anda.
- Dual-layer enkripsi sisi server dengan AWS KMS kunci (DSSE-KMS) — enkripsi Dual-layer sisi server dengan AWS KMS keys (DSSE-KMS) mirip dengan SSE-KMS, tetapi menerapkan dua lapisan individual enkripsi tingkat objek alih-alih satu lapisan.
- Server-side enkripsi dengan kunci yang disediakan pelanggan (SSE-C) — Dengan SSE-C, Anda mengelola kunci enkripsi, dan Amazon S3 mengelola enkripsi saat menulis ke disk, dan dekripsi saat Anda mengakses objek Anda.

Note

Jika Anda memiliki enkripsi sisi server dengan kunci (SSE-C) yang disediakan pelanggan yang diblokir untuk bucket tujuan umum Anda, Anda akan mendapatkan kesalahan HTTP

403 Access Denied saat Anda menentukan header SSE-C permintaan saat menulis data baru ke bucket Anda. Untuk informasi selengkapnya, lihat [Mem blokir atau membuka blokir SSE-C untuk bucket](#) tujuan umum.

Untuk informasi selengkapnya, lihat [Melindungi data menggunakan enkripsi sisi server](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Bergantung pada jenis enkripsi sisi server yang ingin Anda gunakan, tentukan bidang formulir berikut.

- Gunakan SSE-S3, SSE-KMS, atau DSSE-KMS — Jika Anda ingin menggunakan jenis enkripsi sisi server ini, tentukan bidang formulir berikut dalam permintaan.

Nama	Deskripsi	Diperlukan
x-amz-server-side-encryption	Menentukan algoritma enkripsi sisi server yang akan digunakan saat Amazon S3 membuat objek. Untuk menggunakan SSE-S3, tentukan <code>AES256</code> . Untuk menggunakan SSE-KMS, tentukan <code>aws:kms</code> . Untuk menggunakan DSSE-KMS, tentukan <code>aws:kms:dsse</code> . Tipe: String Nilai yang Valid: <code>aws:kms</code> , <code>AES256</code> , <code>aws:kms:dsse</code>	Ya
x-amz-server-side-encryption-aws-kms-key-id	Jika x-amz-server-side-encryption header memiliki nilai valid <code>aws:kms</code> atau <code>aws:kms:dsse</code> , header ini menentukan ID AWS KMS key yang digunakan untuk mengenkripsi objek. Tipe: String	Ya, jika nilainya x-amz-server-side-encryption adalah <code>aws:kms</code> atau

Nama	Deskripsi	Diperlukan
		<code>aws:kms:dsse</code>
<code>x-amz-server-side-encryption-context</code>	Jika <code>x-amz-server-side-encryption</code> memiliki nilai <code>aws:kms</code> atau yang valid <code>aws:kms:dsse</code>, header ini menentukan konteks enkripsi untuk objek. Nilai header ini adalah UTF-8 string yang dikodekan base64 yang berisi pasangan kunci JSON-formatted -nilai untuk konteks enkripsi. Tipe: String	Tidak
<code>x-amz-server-side-encryption-bucket-key-enabled</code>	Jika <code>x-amz-server-side-encryption</code> memiliki nilai valid <code>aws:kms</code> atau <code>aws:kms:dsse</code>, header ini menentukan apakah Amazon S3 harus menggunakan S3 Bucket Key dengan SSE-KMS atau DSSE-KMS. Menyetel header ini untuk <code>true</code> menyebabkan Amazon S3 menggunakan S3 Bucket Key untuk enkripsi objek dengan SSE-KMS atau DSSE-KMS. Jenis: Boolean	Tidak

Note

Jika Anda menentukan `x-amz-server-side-encryption:aws:kms` atau `x-amz-server-side-encryption:aws:kms:dsse`, tetapi tidak memberikan `x-amz-server-side-encryption-aws-kms-key-id`, Amazon S3 menggunakan Kunci yang dikelola AWS (`aws/S3`) untuk melindungi data.

- Gunakan SSE-C — Jika Anda ingin mengelola kunci enkripsi Anda sendiri, Anda harus memberikan semua bidang formulir berikut dalam permintaan.

 Note

Jika Anda menggunakan SSE-C, ETag nilai yang dikembalikan Amazon S3 dalam respons bukanlah MD5 objek.

Nama	Deskripsi	Diperlukan
<code>x-amz-server-side-encryption-customer-algorithm</code>	Menentukan algoritma yang akan digunakan saat mengenkripsi objek. Tipe: String Default: Tidak Ada Nilai yang Valid: AES256 Kendala: Harus disertai dengan <code>x-amz-server-side-encryption-customer-key-MD5</code> bidang <code>x-amz-server-side-encryption-customer-key</code> dan yang valid.	Ya

Nama	Deskripsi	Diperlukan
<code>x-amz-server-side-encryption-customer-key</code>	Menentukan kunci enkripsi base64 yang disediakan pelanggan untuk Amazon S3 untuk digunakan dalam mengenkripsi data. Nilai ini digunakan untuk menyimpan objek, dan kemudian dibuang. Amazon tidak menyimpan kunci enkripsi. Kunci harus sesuai untuk digunakan dengan algoritma yang ditentukan dalam <code>x-amz-server-side-encryption-customer-algorithm</code> header. Tipe: String Default: Tidak Ada Kendala: Harus disertai dengan <code>x-amz-server-side-encryption-customer-key-MD5</code> bidang <code>x-amz-server-side-encryption-customer-algorithm</code> dan yang valid.	Ya
<code>x-amz-server-side-encryption-customer-key-MD5</code>	Menentukan ringkasan MD5 128-bit yang dikodekan base64 dari kunci enkripsi sesuai dengan RFC 1321. http://tools.ietf.org/html/rfc1321 Amazon S3 menggunakan header ini untuk pemeriksaan integritas pesan untuk memastikan bahwa kunci enkripsi ditransmisikan tanpa kesalahan. Tipe: String Default: Tidak Ada Kendala: Harus disertai dengan <code>x-amz-server-side-encryption-customer-key</code> bidang <code>x-amz-server-side-encryption-customer-algorithm</code> dan yang valid.	Ya

Respons

Header Respons

Implementasi operasi ini dapat mencakup header respons berikut selain header respons yang umum untuk semua tanggapan. Untuk informasi selengkapnya, lihat [Header Respons Umum](#).

Nama	Deskripsi
<code>x-amz-checksum-crc32</code>	Checksum objek CRC32 32-bit yang dikodekan base64. Tipe: String
<code>x-amz-checksum-crc32c</code>	Checksum objek CRC32C 32-bit yang dikodekan base64. Tipe: String
<code>x-amz-checksum-sha1</code>	Intisari objek 160-bit yang dikodekan base64 SHA-1 . Tipe: String
<code>x-amz-checksum-sha256</code>	Intisari objek 256-bit yang dikodekan base64 SHA-256 . Tipe: String
<code>x-amz-expiration</code>	Jika <code>Expiration</code> tindakan dikonfigurasi untuk objek sebagai bagian dari konfigurasi siklus hidup bucket, Amazon S3 mengembalikan header ini. Nilai header mencakup <code>expiry-date</code> komponen dan URL-encoded <code>rule-id</code> komponen. Untuk bucket berkemampuan versi, header ini hanya berlaku untuk versi saat ini. Amazon S3 tidak menyediakan header untuk menunjukkan kapan versi noncurrent memenuhi syarat untuk penghapusan permanen. Untuk informasi selengkapnya, lihat PutBucketLifecycleConfiguration . Tipe: String

Nama	Deskripsi
<code>success_action_redirect, redirect</code>	URL tempat klien diarahkan pada unggahan yang berhasil. Tipe: String Leluhur: PostResponse
<code>x-amz-server-side-encryption</code>	Algoritma enkripsi sisi server yang digunakan saat menyimpan objek ini di Amazon S3 (misalnya,, AES256aws:kms,aws:kms:dsse). Tipe: String
<code>x-amz-server-side-encryption-aws-kms-key-id</code>	Jika <code>x-amz-server-side-encryption</code> header memiliki nilai validaws:kms, header ini menentukan ID kunci KMS yang digunakan untuk mengenkripsi objek. Tipe: String
<code>x-amz-server-side-encryption-bucket-key-enabled</code>	Jika <code>x-amz-server-side-encryption</code> memiliki nilai validaws:kms, header ini menunjukkan apakah objek dienkripsi SSE-KMS dengan menggunakan S3 Bucket Key. Jika header ini disetel ketruue, objek menggunakan S3 Bucket Key dengan SSE-KMS. Jenis: Boolean
<code>x-amz-server-side-encryption-customer-algorithm</code>	Jika SSE-C diminta, respons menyertakan header ini, yang mengkonfirmasi algoritma enkripsi yang digunakan. Tipe: String Nilai yang Valid: AES256

Nama	Deskripsi
<code>x-amz-server-side-encryption-customer-key-MD5</code>	Jika SSE-C diminta, respons menyertakan header ini untuk memverifikasi integritas pesan pulang-pergi dari kunci enkripsi yang disediakan pelanggan. Tipe: String
<code>x-amz-version-id</code>	Versi objek. Tipe: String

Elemen Respons

Nama	Deskripsi
Bucket	Nama ember tempat objek disimpan. Tipe: String Leluhur: PostResponse
ETag	Tag entitas (ETag) adalah hash MD5 dari objek yang dapat Anda gunakan untuk melakukan GET operasi bersyarat dengan menggunakan tag <code>If-Modified</code> permintaan dengan operasi GET permintaan. ETag mencerminkan perubahan hanya pada isi objek, bukan pada metadatanya. Tipe: String Leluhur: PostResponse
Key	Nama kunci objek. Tipe: String Leluhur: PostResponse
Location	URI dari objek.

Nama	Deskripsi
	Tipe: String
	Leluhur: PostResponse

Kesalahan Khusus

Implementasi operasi ini tidak mengembalikan kesalahan khusus. Untuk informasi umum tentang kesalahan Amazon S3 dan daftar kode kesalahan, lihat Respons [Kesalahan](#).

Contoh

Permintaan Sampel

```
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

Sampel Respon dengan Pembuatan Versi Ditangguhkan

Berikut ini adalah contoh respons saat pembuatan versi bucket ditangguhkan:

```
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
```

Dalam respons ini, ID versi adalah `null`.

Contoh Respon dengan Versioning Diaktifkan

Berikut ini adalah contoh respons saat pembuatan versi bucket diaktifkan.

```
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
```

Sumber Daya Terkait

- [CopyObject](#)
- [Objek POST](#)
- [GetObject](#)

Pemulihan objek POST

Deskripsi

Operasi ini melakukan jenis permintaan berikut:

- `select`— Lakukan kueri pilih pada objek yang diarsipkan
- `restore an archive`— Kembalikan objek yang diarsipkan

Untuk menggunakan operasi ini, Anda harus memiliki izin untuk melakukan `s3:GetObject` tindakan `s3:RestoreObject` dan. Pemilik bucket memiliki izin ini secara default dan dapat memberikan izin ini kepada orang lain. Untuk informasi selengkapnya tentang izin, lihat [Izin Terkait Operasi Sub-Sumber Daya Bucket](#) dan [Mengelola Izin Akses ke Sumber Daya Amazon S3 Anda](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Mengkueri Arsip dengan Permintaan Pilihan

Anda menggunakan jenis permintaan tertentu untuk melakukan kueri SQL pada objek yang diarsipkan. Objek yang diarsipkan yang sedang ditanyakan oleh permintaan pilih harus diformat sebagai file nilai yang dipisahkan koma (CSV) yang tidak terkompresi. Anda dapat menjalankan kueri dan analitik kustom pada data yang diarsipkan tanpa harus mengembalikan data ke tingkat Amazon S3 yang lebih panas. Untuk ikhtisar tentang permintaan tertentu, lihat Mengkueri [Objek yang Diarsipkan](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Saat membuat permintaan pilih, lakukan hal berikut:

- Tentukan lokasi keluaran untuk output kueri pilih. Ini harus berupa bucket Amazon S3 yang Wilayah AWS sama dengan bucket yang berisi objek arsip yang sedang ditanyakan. Akun AWS Yang memulai pekerjaan harus memiliki izin untuk menulis ke bucket S3. Anda dapat menentukan kelas penyimpanan dan enkripsi untuk objek output yang disimpan dalam bucket. Untuk informasi selengkapnya tentang keluaran, lihat Mengku [eri Objek yang Diarsipkan](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Untuk informasi selengkapnya tentang S3 struktur dalam badan permintaan, lihat berikut ini:

- [PutObject](#)
- [Mengelola Akses dengan ACL](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon

- [Melindungi Data Menggunakan Server-Side Enkripsi](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon
- Tentukan ekspresi SQL untuk SELECT jenis pemulihan untuk kueri Anda dalam `SelectParameters` struktur badan permintaan. Anda dapat menggunakan ekspresi seperti contoh berikut.
- Ekspresi berikut mengembalikan semua catatan dari objek yang ditentukan.

```
SELECT * FROM Object
```

- Dengan asumsi bahwa Anda tidak menggunakan header apa pun untuk data yang disimpan dalam objek, Anda dapat menentukan kolom dengan header posisi.

```
SELECT s._1, s._2 FROM Object s WHERE s._3 > 100
```

- Jika Anda memiliki header dan Anda mengatur CSV struktur `fileHeaderInfo` di badan permintaan ke `USE`, Anda dapat menentukan header dalam kueri. (Jika Anda mengatur bidang `fileHeaderInfo` ke `IGNORE`, baris pertama dilewati untuk kueri.) Anda tidak dapat mencampur posisi ordinal dengan nama kolom header.

```
SELECT s.Id, s.FirstName, s.SSN FROM S3Object s
```

Untuk informasi selengkapnya tentang menggunakan SQL dengan pemulihan Amazon Glacier Select, lihat Referensi [SQL untuk Amazon S3 Select dan Amazon Glacier Select](#) di Panduan Pengguna Amazon Simple Storage Service.

Saat membuat permintaan pilih, Anda juga dapat melakukan hal berikut:

- Untuk mempercepat kueri Anda, tentukan tingkatnya. `Expedited` Untuk informasi selengkapnya tentang tingkatan, lihat “Memulihkan Arsip,” nanti dalam topik ini.
- Tentukan detail tentang format serialisasi data dari objek input yang sedang ditanyakan dan serialisasi hasil kueri. `CSV-encoded`

Berikut ini adalah fakta penting tambahan tentang fitur pilih:

- Hasil output adalah objek Amazon S3 baru. Tidak seperti pengambilan arsip, mereka disimpan hingga dihapus secara eksplisit — secara manual atau melalui kebijakan siklus hidup.

- Anda dapat mengeluarkan lebih dari satu permintaan pilih pada objek Amazon S3 yang sama. Amazon S3 tidak mendeduplikasi permintaan, jadi hindari mengeluarkan permintaan duplikat.
- Amazon S3 menerima permintaan pilih meskipun objek telah dipulihkan. Permintaan pilih tidak mengembalikan respons kesalahan 409.

Memulihkan Arsip

Objek di kelas penyimpanan GLACIER dan DEEP_ARCHIVE diarsipkan. Untuk mengakses objek yang diarsipkan, Anda harus terlebih dahulu memulai permintaan pemulihan. Ini mengembalikan salinan sementara dari objek yang diarsipkan. Dalam permintaan pemulihan, Anda menentukan jumlah hari di mana Anda ingin salinan yang dipulihkan ada. Setelah periode yang ditentukan, Amazon S3 menghapus salinan sementara tetapi objek tetap diarsipkan di kelas penyimpanan GLACIER atau DEEP_ARCHIVE tempat objek dipulihkan.

Untuk mengembalikan versi objek tertentu, Anda dapat memberikan ID versi. Jika Anda tidak memberikan ID versi, Amazon S3 mengembalikan versi saat ini.

Waktu yang dibutuhkan pekerjaan pemulihan untuk menyelesaikan tergantung pada kelas penyimpanan objek yang dipulihkan dan tingkat akses data yang Anda tentukan.

Saat memulihkan objek yang diarsipkan (atau menggunakan permintaan pilih), Anda dapat menentukan salah satu opsi tingkat akses data berikut di `Tier` elemen badan permintaan:

- **Expedited**- Pengambilan yang dipercepat memungkinkan Anda untuk dengan cepat mengakses data Anda yang disimpan di kelas penyimpanan GLACIER ketika permintaan mendesak sesekali untuk subset arsip diperlukan. Untuk semua kecuali objek yang diarsipkan terbesar (250 MB +), data yang diakses menggunakan pengambilan cepat biasanya tersedia dalam 1—5 menit. Kapasitas yang disediakan memastikan bahwa kapasitas pengambilan untuk pengambilan yang dipercepat tersedia saat Anda membutuhkannya. Pengambilan yang dipercepat dan kapasitas yang disediakan tidak tersedia untuk kelas penyimpanan DEEP_ARCHIVE.
- **Standard**- Pengambilan standar memungkinkan Anda untuk mengakses salah satu objek yang diarsipkan dalam beberapa jam. Ini adalah opsi default untuk permintaan pengambilan GLACIER dan DEEP_ARCHIVE yang tidak menentukan opsi pengambilan. Pengambilan standar biasanya selesai dalam waktu 3-5 jam dari kelas penyimpanan GLACIER dan biasanya selesai dalam waktu 12 jam dari kelas penyimpanan DEEP_ARCHIVE.
- **Bulk**- Pengambilan massal adalah opsi pengambilan biaya terendah Amazon Glacier, memungkinkan Anda mengambil data dalam jumlah besar, bahkan petabyte, dengan harga murah

dalam sehari. Pengambilan massal biasanya selesai dalam 5-12 jam dari kelas penyimpanan GLACIER dan biasanya selesai dalam waktu 48 jam dari kelas penyimpanan DEEP_ARCHIVE.

Untuk informasi selengkapnya tentang opsi pengambilan arsip dan kapasitas yang disediakan untuk akses Expedited data, lihat Memulihkan Objek [yang Diarsipkan](#) di Panduan Pengguna Amazon Simple Storage Service.

Anda dapat menggunakan peningkatan kecepatan pemulihan Amazon S3 untuk mengubah kecepatan pemulihan ke kecepatan yang lebih cepat saat sedang berlangsung. Anda meningkatkan kecepatan pemulihan yang sedang berlangsung dengan mengeluarkan permintaan pemulihan lain ke objek yang sama, menyetel elemen Tier permintaan baru. Ketika mengeluarkan permintaan untuk meningkatkan tingkat pemulihan, Anda harus memilih tingkat yang lebih cepat daripada tingkat yang sedang digunakan untuk pemulihan. Anda tidak boleh mengubah parameter lain, seperti elemen permintaan Days. Untuk informasi selengkapnya, lihat Meng [upgrade Kecepatan Pem In-Progress ulihan](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Untuk mendapatkan status pemulihan objek, Anda dapat mengirim HEAD permintaan. Operasi mengembalikan x-amz-restore header, yang memberikan informasi tentang status pemulihan, dalam respons. Anda dapat menggunakan notifikasi peristiwa Amazon S3 untuk memberi tahu Anda saat pemulihan dimulai atau selesai. Untuk informasi selengkapnya, lihat Meng [onfigurasi Pemberitahuan Peristiwa Amazon S3](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Setelah memulihkan objek yang diarsipkan, Anda dapat memperbarui periode pemulihan dengan menerbitkan kembali permintaan dengan periode baru. Amazon S3 memperbarui periode pemulihan relatif terhadap waktu saat ini dan hanya mengenakan biaya untuk permintaan—tidak ada biaya transfer data. Anda tidak dapat memperbarui periode pemulihan saat Amazon S3 secara aktif memproses permintaan pemulihan Anda saat ini untuk objek tersebut.

Jika bucket Anda memiliki konfigurasi siklus hidup dengan aturan yang menyertakan tindakan kedaluwarsa, kedaluwarsa objek menggantikan rentang hidup yang Anda tentukan dalam permintaan pemulihan. Misalnya, jika Anda memulihkan salinan objek selama 10 hari, tetapi objek dijadwalkan kedaluwarsa dalam 3 hari, Amazon S3 menghapus objek dalam 3 hari. Untuk informasi selengkapnya tentang konfigurasi siklus hidup, lihat [PutBucketLifecycleConfiguration](#) dan [Manajemen Siklus Hidup Ob](#) jek di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Permintaan

Sintaksis

```
POST /ObjectName?restore&versionId=VersionID HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Permintaan Otentikasi \(AWS Versi Tanda Tangan 4\))
Content-MD5: MD5

request body
```

Note

Sintaks menunjukkan beberapa header permintaan. Untuk daftar lengkapnya, lihat “Minta Header,” nanti dalam topik ini.

Parameter Permintaan

Implementasi operasi ini tidak menggunakan parameter permintaan.

Header Permintaan

Nama	Deskripsi	Diperlukan
Content-MD5	Intisari data MD5 128-bit yang dikodekan base64. Anda harus menggunakan header ini sebagai pemeriksaan integritas pesan untuk memverifikasi bahwa badan permintaan tidak rusak saat transit. Untuk informasi lebih lanjut, lihat RFC 1864. Tipe: String Default: Tidak Ada	Ya

Elemen permintaan

Berikut ini adalah contoh XML dari badan permintaan untuk memulihkan arsip.

```
<RestoreRequest>
  <Days>2</Days>
  <GlacierJobParameters>
    <Tier>Bulk</Tier>
  </GlacierJobParameters>
</RestoreRequest>
```

Tabel berikut menjelaskan XML untuk pemulihan arsip di badan permintaan.

Nama	Deskripsi	Diperlukan
RestoreRequest	Wadah untuk memulihkan informasi. Tipe: Kontainer	Ya
Days	Seumur hidup salinan yang dipulihkan (aktif). Jumlah hari minimum Anda dapat memulihkan objek dari Amazon Glacier adalah 1. Setelah salinan objek mencapai masa pakai yang ditentukan, Amazon S3 menghapusnya dari bucket. Jika Anda memulihkan arsip, elemen ini diperlukan. Jangan gunakan elemen ini dengan SELECT jenis permintaan. Jenis: Integer positif Leluhur: RestoreRequest	Ya, jika memulihkan arsip
GlacierJobParameters	Wadah untuk parameter pekerjaan kelas penyimpanan S3 Glacier. Jangan gunakan elemen ini dengan SELECT jenis permintaan. Tipe: Kontainer	Tidak

Nama	Deskripsi	Diperlukan
	Nenek moyang: RestoreRequest	
Tier	Tingkat akses data yang akan digunakan saat memulihkan arsip. Standard adalah default. Jenis: Enum Nilai valid: Expedited Standard Bulk Nenek moyang: GlacierJobParameters	Tidak

XML berikut adalah badan permintaan untuk kueri pilih pada objek yang diarsipkan:

```
<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>
</RestoreRequest>
```

```

</SelectParameters>
<OutputLocation>
  <S3>
    <BucketName>Name of bucket</BucketName>
    <Prefix>Key prefix</Prefix>
    <CannedACL>Canned ACL string</CannedACL>
    <AccessControlList>
      <Grantee>
        <Type>Grantee Type</Type>
      </Grantee>
      <ID>Grantee identifier</ID>
      <URI>Grantee URI</URI>
      <Permission>Granted permission</Permission>
      <DisplayName>Display Name</DisplayName>
      <EmailAddress>email</EmailAddress>
    </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>

```

Tabel berikut menjelaskan XML untuk SELECT jenis restorasi dalam badan permintaan.

Nama	Deskripsi	Diperlukan
RestoreRequest	Wadah untuk memulihkan informasi. Tipe: Kontainer	Ya
Tier	Tingkat akses data yang akan digunakan saat memulihkan arsip. Standard adalah default. Jenis: Enum Nilai valid: Expedited Standard Bulk Nenek moyang: RestoreRequest	Tidak
Description	Deskripsi opsional untuk permintaan. Tipe: String Nenek moyang: RestoreRequest	Tidak
SelectParameters	Menjelaskan parameter untuk permintaan pekerjaan yang dipilih. Tipe: Kontainer Nenek moyang: RestoreRequest	Ya, jika jenis permintaan adalah SELECT
OutputLocation	Menjelaskan lokasi yang menerima hasil permintaan pemulihan pilih. Jenis: Wadah untuk Amazon S3 Leluhur: RestoreRequest	Ya, jika jenis permintaan adalah SELECT

Elem **SelectParameters** en wadah berisi elemen-elemen berikut.

Nama	Deskripsi	Diperlukan
Expression	Ekspresi SQL. Contoh: - Ekspresi SQL berikut mengambil kolom pertama data dari objek yang disimpan dalam format CSV: <pre>SELECT s._1 FROM Object s</pre> - Ekspresi SQL berikut mengembalikan semuanya dari objek: <pre>SELECT * FROM Object</pre> Tipe: String Leluhur: SelectParameters	Ya
ExpressionType	Mengidentifikasi jenis ekspresi. Tipe: String Nilai yang valid: SQL Leluhur: SelectParameters	Ya
InputSerialization	Menjelaskan format serialisasi objek. Jenis: Wadah untuk CSV Leluhur: SelectParameters	Ya
OutputSerialization	Menjelaskan bagaimana hasil dari tugas pilih diserialkan. Jenis: Wadah untuk CSV	Ya

Nama	Deskripsi	Diperlukan
	Leluhur: <code>SelectParameters</code>	

Elemen penampung CSV dalam **InputSerialization** elemen berisi elemen-elemen berikut.

Nama	Deskripsi	Diperlukan
RecordDelimiter	Karakter tunggal digunakan untuk memisahkan catatan individu dalam input. Alih-alih nilai default, Anda dapat menentukan pembatas arbitrer. Tipe: String Default: <code>\n</code> Leluhur: CSV	Tidak
FieldDelimiter	Karakter tunggal digunakan untuk memisahkan bidang individual dalam catatan. Anda dapat menentukan pembatas sewenang-wenang. Tipe: String Default: <code>,</code> Leluhur: CSV	Tidak
QuoteCharacter	Karakter tunggal yang digunakan untuk melarikan diri ketika pembatas bidang adalah bagian dari nilai. Pertimbangkan contoh ini dalam file CSV: <code>"a, b"</code> Membungkus nilai dalam tanda kutip membuat nilai ini menjadi satu bidang. Jika Anda tidak menggunakan tanda	Tidak

Nama	Deskripsi	Diperlukan
	kutip, koma adalah pembatas bidang (yang menjadikannya dua nilai bidang terpisah, a dan b). Tipe: String Default: " Leluhur: CSV	
QuoteEscapeCharacter	Karakter tunggal yang digunakan untuk melarikan diri dari karakter tanda kutip di dalam nilai yang sudah lolos. Misalnya, nilainya "" a , b "" diurai sebagai" a , b ". Tipe: String Default: " Leluhur: CSV	Tidak

Nama	Deskripsi	Diperlukan
FileHeaderInfo	Menjelaskan baris pertama dalam data input. Ini adalah salah satu nilai ENUM. • NONE: Baris pertama bukan header. • IGNORE: Baris pertama adalah header, tetapi Anda tidak dapat menggunakan nilai header untuk menunjukkan kolom dalam ekspresi. Anda dapat menggunakan posisi kolom (seperti <code>_1</code>, <code>_2</code>,...) untuk menunjukkan kolom (<code>SELECT s._1 FROM OBJECT s</code>). • Use: Baris pertama adalah header, dan Anda dapat menggunakan nilai header untuk mengidentifikasi kolom dalam ekspresi (<code>SELECT "name" FROM OBJECT</code>). Jenis: Enum Nilai yang valid: NONE USE IGNORE Leluhur: CSV	Tidak
Comments	Karakter tunggal yang digunakan untuk menunjukkan baris harus diabaikan ketika karakter muncul di awal baris tersebut. Anda dapat menentukan karakter apa pun untuk menunjukkan baris komentar. Tipe: String Leluhur: CSV	Tidak

Elemen penampung CSV (dalam **OutputSerialization** elemen) berisi elemen-elemen berikut.

Nama	Deskripsi	Diperlukan
QuoteFields	Menunjukkan apakah akan menggunakan tanda kutip di sekitar bidang keluaran. • ALWAYS: Selalu gunakan tanda kutip untuk bidang keluaran. • ASNEEDED: Gunakan tanda kutip untuk bidang keluaran bila diperlukan. Jenis: Enum Nilai yang valid: ALWAYS ASNEEDED Default: AsNeeded Leluhur: CSV	Tidak
RecordDelimiter	Karakter tunggal digunakan untuk memisahkan catatan individu dalam output. Alih-alih nilai default, Anda dapat menentukan pembatas arbitrer. Tipe: String Default: \n Leluhur: CSV	Tidak
FieldDelimiter	Karakter tunggal digunakan untuk memisahkan bidang individual dalam catatan. Anda dapat menentukan pembatas sewenang-wenang. Tipe: String	Tidak

Nama	Deskripsi	Diperlukan
	Default: , Leluhur: CSV	
QuoteCharacter	Karakter tunggal yang digunakan untuk melarikan diri ketika pembatas bidang adalah bagian dari nilai. Misalnya, jika nilainya adalah , b, Amazon S3 membungkus nilai bidang ini dalam tanda kutip, sebagai berikut: " a , b " . Tipe: String Default: " Leluhur: CSV	Tidak
QuoteEscapeCharacter	Karakter tunggal yang digunakan untuk melarikan diri dari karakter tanda kutip di dalam nilai yang sudah lolos. Misalnya, jika nilainya adalah " a , b " , Amazon S3 membungkus nilai dalam tanda kutip, sebagai berikut: "" " a , b "" " . Tipe: String Leluhur: CSV	Tidak

Elemen wadah S3 (dalam **OutputLocation** elemen) berisi elemen-elemen berikut.

Nama	Deskripsi	Diperlukan
AccessControlList	Daftar hibah yang mengontrol akses ke hasil bertahap. Jenis: Wadah untuk Hibah Leluhur: S3	Tidak
BucketName		Ya

Nama	Deskripsi	Diperlukan
	Nama bucket S3 tempat hasil pemulihan pilih disimpan. Bucket harus Wilayah AWS sama dengan bucket yang berisi objek arsip input. Tipe: String Leluhur: S3	
CannedACL	Daftar kontrol akses terkaleng (ACL) untuk diterapkan pada hasil pemulihan yang dipilih. Tipe: String Nilai yang valid: <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> Leluhur: S3	Tidak
Encryption	Berisi informasi enkripsi untuk hasil yang disimpan. Jenis: Wadah untuk Enkripsi Leluhur: S3	Tidak
Prefix	Awalan yang ditambahkan ke hasil pemulihan pilih. Panjang maksimum prefiks adalah 512 byte. Tipe: String Leluhur: S3	Ya

Nama	Deskripsi	Diperlukan
StorageClass	Kelas penyimpanan yang digunakan untuk menyimpan hasil permintaan pilih. Tipe: String Nilai valid: STANDARD REDUCED_REDUNDANCY STANDARD_IA ONEZONE_IA Leluhur: S3	Tidak
Tagging	Wadah untuk informasi tag. Jenis: Struktur tag Leluhur: S3	Tidak
UserMetadata	Berisi daftar metadata untuk disimpan dengan hasil pemulihan pilih. Jenis: MetadataEntry struktur Leluhur: S3	Tidak

Elemen kontainer Grantee (dalam **AccessControlList** elemen) berisi elemen-elemen berikut.

Nama	Deskripsi	Diperlukan
DisplayName	Nama layar penerima izin. Tipe: String Leluhur: Grantee	Tidak
EmailAddress	Alamat email penerima izin.	Tidak

Nama	Deskripsi	Diperlukan
ID	Tipe: String	Tidak
	Leluhur: Grantee	
	ID pengguna kanonik penerima izin.	
Type	Tipe: String	Tidak
	Leluhur: Grantee	
	Tipe penerima izin.	
URI	Tipe: String	Tidak
	Leluhur: Grantee	
	URI dari grup penerima izin.	
Permission	Tipe: String	Tidak
	Leluhur: Grantee	
	Diberikan izin.	

Elemen wadah Enkripsi (dalam S3) berisi elemen-elemen berikut.

Nama	Deskripsi	Diperlukan
EncryptionType	Algoritma enkripsi sisi server yang digunakan saat menyimpan hasil pekerjaan. Defaultnya adalah tidak ada enkripsi.	Tidak

Nama	Deskripsi	Diperlukan
	Tipe: String Nilai yang Valid aws : kms AES256 Leluhur: Encryption	
KMSContext	Tidak wajib. Jika jenis enkripsi adalah aws : kms, Anda dapat menggunakan nilai ini untuk menentukan konteks enkripsi untuk hasil pemulihan yang dipilih. Tipe: String Leluhur: Enkripsi	Tidak
KMSKeyId	ID kunci AWS Key Management Service (AWS KMS) untuk digunakan untuk enkripsi objek. Tipe: String Leluhur: Encryption	Tidak

Elem **TagSet** adalah wadah (dalam **elemen** Tagging) berisi elemen berikut.

Nama	Deskripsi	Diperlukan
Tag	Berisi tag. Tipe: Kontainer Leluhur: TagSet	Tidak

Elemen wadah Tag (dalam **TagSet** elemen) berisi elemen-elemen berikut.

Nama	Deskripsi	Diperlukan
Key		Tidak

Nama	Deskripsi	Diperlukan
	Nama tanda. Tipe: String Leluhur: Tag	
Value	Nilai tag. Tipe: String Leluhur: Tag	Tidak

Elemen **MetadataEntry** wadah (dalam **UserMetadata** elemen) berisi elemen pasangan kunci-nilai berikut untuk disimpan dengan objek.

Nama	Deskripsi	Diperlukan
MetadataKey	Kunci metadata. Tipe: String Leluhur:	Tidak
MetadataEntry	Nilai metadata. Tipe: String Leluhur:	Tidak

Respons

Operasi yang berhasil mengembalikan kode 202 Accepted status 200 OK atau.

- Jika salinan objek sebelumnya tidak dipulihkan, maka Amazon S3 kembali 202 Accepted dalam respons.
- Jika salinan objek dipulihkan sebelumnya, Amazon S3 kembali 200 OK dalam respons.

Header Respons

Implementasi operasi ini hanya menggunakan header respons yang umum untuk sebagian besar respons. Untuk informasi selengkapnya, lihat [Header Respons Umum](#).

Elemen Respons

Operasi ini tidak mengembalikan elemen respons.

Kesalahan Khusus

Kode Kesalahan	Deskripsi	Kode Status HTTP	Awalan Kode Kesalahan SOAP
RestoreAlreadyInProgress	Pemulihan objek sudah berlangsung. (Kesalahan ini tidak berlaku untuk permintaan tipe SELECT.)	409 Konflik	Klien
GlacierExpeditedRetrievalNotAvailable	Pengambilan dipercepat kelas penyimpanan S3 Glacier saat ini tidak tersedia. Coba lagi nanti. (Dikembalikan jika kapasitas tidak mencukupi untuk memproses Expedited permintaan. Kesalahan ini hanya berlaku untuk Expedited pengambilan dan bukan untuk Standard atau pengambilan.Bulk)	503	N/A

Contoh

Kembalikan Objek selama Dua Hari Menggunakan Opsi Pengambilan Dipercepat

Permintaan pemulihan berikut mengembalikan salinan photo1.jpg objek dari Amazon Glacier selama dua hari menggunakan opsi pengambilan yang dipercepat.

```
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>
```

Jika examplebucket tidak memiliki salinan objek yang dipulihkan, Amazon S3 mengembalikan 202 Accepted respons berikut.

```
HTTP/1.1 202 Accepted
x-amz-id-2: GFihv3y6+kE7KG11GEkQhU7/2/cHR3Yb2fCb2S04nxI423Dqwg2XiQ0B/
UZlZyQvPiBlZNRcovw=
x-amz-request-id: 9F341CD3C4BA79E0
Date: Sat, 20 Oct 2012 23:54:05 GMT
Content-Length: 0
Server: AmazonS3
```

Jika salinan objek sudah dipulihkan, Amazon S3 mengembalikan 200 OK respons, dan hanya memperbarui waktu kedaluwarsa salinan yang dipulihkan.

Kueri Arsip dengan Permintaan SELECT

Berikut ini adalah contoh pilih permintaan pemulihan.

```
POST /object-one.csv?restore HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: 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>
  </S3>
</OutputLocation>

```

```
</UserMetadata>
<StorageClass>STANDARD</StorageClass>
</S3>
</OutputLocation>
</RestoreRequest>
```

Amazon S3 mengembalikan 202 Accepted respons berikut.

```
HTTP/1.1 202 Accepted
x-amz-id-2: GFihv3y6+kE7KG11GEkQhU7/2/cHR3Yb2fCb2S04nxI423Dqwg2XiQ0B/
UZlZYQvPiBlZNRcovw=
x-amz-request-id: 9F341CD3C4BA79E0
x-amz-restore-output-path: js-test-s3/qE8nk5M0XIj-LuZE2HXNw6empQm3znLkHlMWInRYPS-
0r12W0uj6LyYm-neTvm1-btz3wbBxfMhPykd3jk1-lvZE7w42/
Date: Sat, 20 Oct 2012 23:54:05 GMT
Content-Length: 0
Server: AmazonS3
```

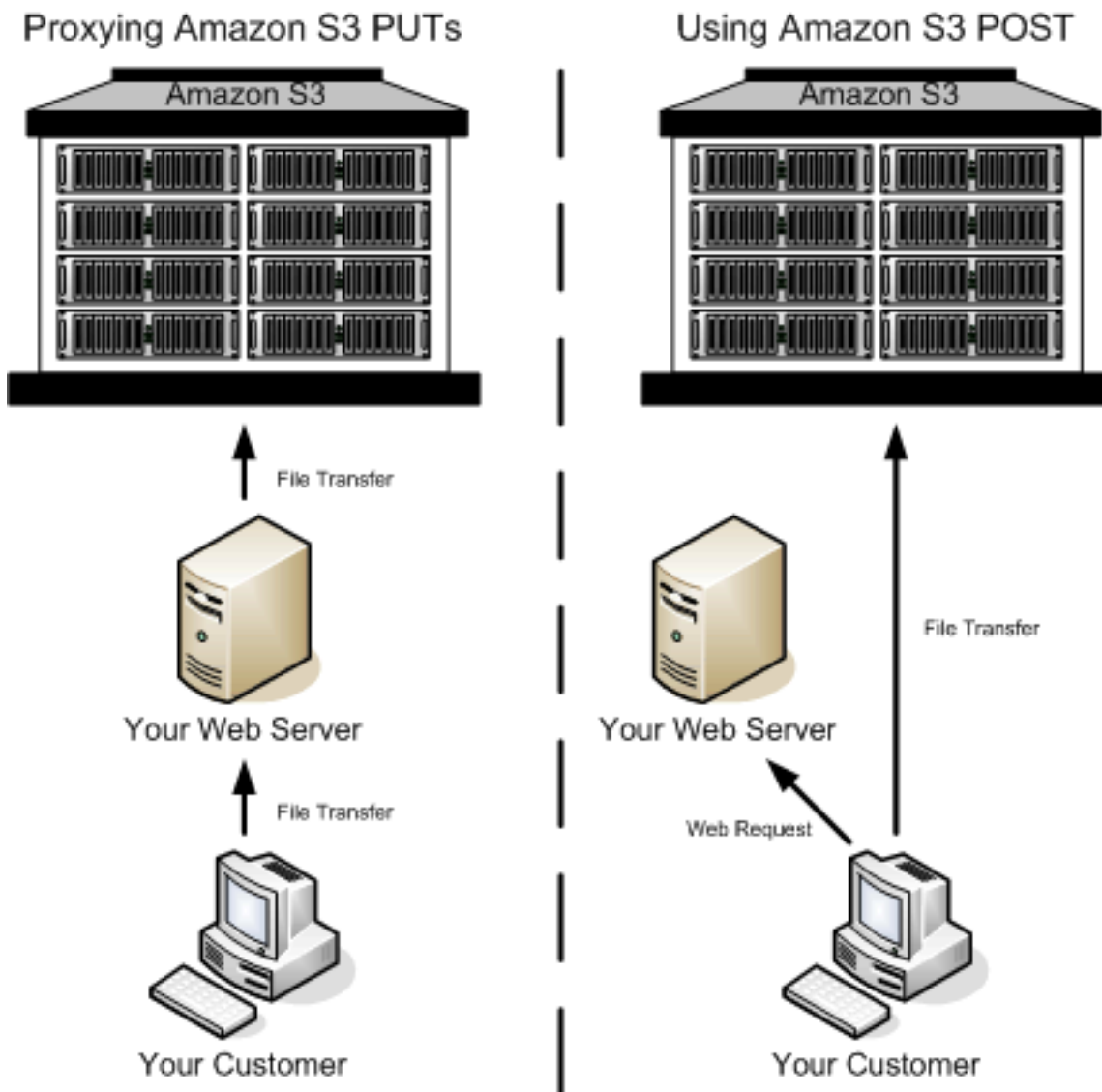
Info Selengkapnya

- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)
- [Referensi SQL untuk Amazon S3 Select dan Amazon Glacier Select](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon

Browser-Based Unggah Menggunakan HTTP POST

Amazon S3 mendukung permintaan HTTP POST sehingga pengguna dapat mengunggah konten langsung ke Amazon S3. Dengan menggunakan POST, pengguna akhir dapat mengotentikasi permintaan tanpa harus meneruskan data melalui node perantara aman yang melindungi kredensial Anda. Dengan demikian, HTTP POST memiliki potensi untuk mengurangi latensi.

Gambar berikut menunjukkan unggahan Amazon S3 menggunakan permintaan POST.



1. Pengguna mengakses halaman Anda dari browser web.
2. Halaman web Anda berisi formulir HTML yang berisi semua informasi yang diperlukan pengguna untuk mengunggah konten ke Amazon S3.
3. Pengguna mengunggah konten ke Amazon S3 melalui browser web.

Proses pengiriman permintaan POST berbasis browser adalah sebagai berikut:

1. Buat kebijakan keamanan yang menentukan kondisi yang membatasi apa yang ingin Anda izinkan dalam permintaan, seperti nama bucket tempat objek dapat diunggah, dan awalan nama kunci yang ingin Anda izinkan untuk objek yang sedang dibuat.

2. Buat tanda tangan yang didasarkan pada kebijakan. Untuk permintaan yang diautentikasi, formulir harus menyertakan tanda tangan yang valid dan kebijakan.
3. Buat formulir HTML yang dapat diakses pengguna Anda untuk mengunggah objek ke bucket Amazon S3 Anda.

Bagian berikut menjelaskan cara membuat tanda tangan untuk mengotentikasi permintaan. Untuk informasi tentang membuat formulir dan kebijakan keamanan, lihat [Membuat Formulir HTML \(Menggunakan AWS Versi Tanda Tangan 4\)](#).

Menghitung Tanda Tangan

Untuk permintaan yang diautentikasi, formulir HTML harus menyertakan bidang untuk kebijakan keamanan dan tanda tangan.

- Kebijakan keamanan (lihat [Kebijakan POST](#)) mengontrol apa yang diizinkan dalam permintaan.
- Kebijakan keamanan adalah `StringToSign` (lihat [Pengantar Permintaan Penandatanganan](#)) dalam perhitungan tanda tangan Anda.



Untuk Menghitung Tanda Tangan

1. Buat kebijakan menggunakan UTF-8 pengkodean.
2. Konversi byte UTF-8-encoded kebijakan ke base64. Hasilnya adalah `StringToSign`.
3. Buat kunci penandatanganan.
4. Gunakan kunci penandatanganan untuk menandatangani `StringToSign` menggunakan algoritma HMAC-SHA256 penandatanganan.

Untuk informasi selengkapnya tentang membuat formulir HTML, kebijakan keamanan, dan contoh, lihat berikut ini:

- [Membuat Formulir HTML \(Menggunakan AWS Versi Tanda Tangan 4\)](#)
- [Kebijakan POST](#)
- [Contoh: Browser-Based Unggah menggunakan HTTP POST \(Menggunakan AWS Versi Tanda Tangan 4\)](#)

Membuat Formulir HTML (Menggunakan AWS Versi Tanda Tangan 4)

Topik

- [Deklarasi Formulir HTML](#)
- [Bidang Formulir HTML](#)

Untuk mengizinkan pengguna mengunggah konten ke Amazon S3 dengan menggunakan browser mereka (permintaan HTTP POST), Anda menggunakan formulir HTML. Formulir HTML terdiri dari deklarasi formulir dan bidang formulir. Deklarasi formulir berisi informasi tingkat tinggi mengenai permintaan tersebut. Bidang formulir berisi informasi permintaan terperinci.

Bagian ini menjelaskan cara membuat formulir HTML. Untuk contoh kerja unggahan berbasis browser menggunakan HTTP POST dan perhitungan tanda tangan terkait untuk otentikasi permintaan, lihat [Contoh: Browser-Based Unggah menggunakan HTTP POST \(Menggunakan AWS Versi Tanda Tangan 4\)](#).

Bentuk dan kebijakan harus di UTF-8 kodekan. Anda dapat menerapkan UTF-8 pengkodean ke formulir dengan menentukan `charset=UTF-8` dalam `content` atribut. Berikut ini adalah contoh peng UTF-8 kodean dalam judul HTML.

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

Berikut ini adalah contoh peng UTF-8 kodean dalam header permintaan.

```
Content-Type: text/html; charset=UTF-8
```

Note

Data formulir dan batas (tidak termasuk isi file) tidak boleh melebihi 20KB.

Deklarasi Formulir HTML

Deklarasi formulir HTML memiliki tiga atribut berikut:

- **action**— URL yang memproses permintaan, yang harus disetel ke URL bucket. Misalnya, jika nama bucket Anda adalah `examplebucket`, URL-nya adalah `http://examplebucket.s3.amazonaws.com/`.

Note

Nama kunci ditentukan dalam bidang formulir.

- **method**— Metode harus POST.
- **enctype**- Jenis penutup (`enctype`) harus disetel ke `multipart/form-data` untuk unggahan file dan unggahan area teks. Untuk informasi lebih lanjut tentang `enctype`, lihat [RFC 1867](https://tools.ietf.org/html/rfc1867).

Ini adalah deklarasi formulir untuk ember `examplebucket`.

```
<form action="http://examplebucket.s3.amazonaws.com/" method="post"
enctype="multipart/form-data">
```

Bidang Formulir HTML

Tabel berikut menjelaskan daftar bidang yang dapat Anda gunakan dalam formulir. Di antara bidang lainnya, ada bidang tanda tangan yang dapat Anda gunakan untuk mengotentikasi permintaan. Ada bidang bagi Anda untuk menentukan algoritma perhitungan tanda tangan (`x-amz-algorithm`), cakupan kredensial (`x-amz-credential`) yang Anda gunakan untuk menghasilkan kunci penandatanganan, dan tanggal (`x-amz-date`) yang digunakan untuk menghitung tanda tangan. Amazon S3 menggunakan informasi ini untuk membuat ulang tanda tangan. Jika tanda tangan cocok, Amazon S3 memproses permintaan.

Note

Variabel `${filename}` secara otomatis diganti dengan nama file yang diberikan oleh pengguna dan dikenali oleh semua bidang formulir. Jika browser atau klien menyediakan jalur penuh atau sebagian ke file, hanya teks yang mengikuti garis miring terakhir (/) atau garis miring terbalik (\) yang digunakan (misalnya, `C:\Program Files\directory1\file.txt` ditafsirkan sebagai). `file.txt` Jika tidak ada nama file atau file yang diberikan, maka variabel diganti dengan string kosong.

Jika Anda tidak menyediakan elemen yang diperlukan untuk permintaan yang diautentikasi, seperti `policy` elemen, permintaan diasumsikan anonim dan akan berhasil hanya jika Anda telah mengonfigurasi bucket untuk baca dan tulis publik.

Nama Elemen	Deskripsi	Diperlukan
<code>acl</code>	Daftar kontrol akses (ACL) Amazon S3. Jika ACL tidak valid ditentukan, Amazon S3 menolak permintaan tersebut. Untuk informasi selengkapnya tentang ACL, lihat Menggunakan Amazon S3 ACL. Tipe: String	Tidak

Nama Elemen	Deskripsi	Diperlukan
	Default: privat Nilai Valid: private public-read public-read-write aws-exec-read authenticated-read bucket-owner-read bucket-owner-full-control	
Cache-Control	REST-specific tajuk. Untuk informasi selengkapnya, lihat PutObject .	Tidak
Content-Type		
Content-Disposition		
Content-Encoding		
Expires		
key	Nama kunci dari objek yang diunggah. Untuk menggunakan nama file yang disediakan oleh pengguna, gunakan variabel <code>\${filename}</code>. Misalnya, jika Anda mengunggah file <code>photo1.jpg</code> dan Anda menentukan <code>/user/user1/\${filename}</code> sebagai nama kunci, file tersebut disimpan sebagai <code>user/user1/photo1.jpg</code>. Untuk informasi selengkapnya, lihat Kunci Objek dan Metadata di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.	Ya

Nama Elemen	Deskripsi	Diperlukan
<code>policy</code>	Kebijakan keamanan yang dikodekan base64 yang menjelaskan apa yang diizinkan dalam permintaan. Untuk permintaan yang diautentikasi, kebijakan diperlukan. Permintaan tanpa kebijakan keamanan dianggap anonim dan hanya akan berhasil pada bucket yang dapat ditulis secara publik.	Diperlukan untuk permintaan yang diautentikasi
<code>success_action_redirect</code>	URL yang akan diarahkan klien setelah unggahan berhasil. Jika tidak <code>success_action_redirect</code> ditentukan, atau Amazon S3 tidak dapat menafsirkan URL, Amazon S3 mengembalikan jenis dokumen kosong yang ditentukan di <code>success_action_status</code> bidang. Jika unggahan gagal, Amazon S3 mengembalikan kesalahan dan tidak mengarahkan pengguna ke URL lain.	Tidak

Nama Elemen	Deskripsi	Diperlukan
success_action_status	Kode status dikembalikan ke klien setelah berhasil diunggah jika tidak success_action_redirect ditentukan. Nilai yang valid adalah 200, 201, atau 204 (default). Jika nilai disetel ke 200 atau 204, Amazon S3 mengembalikan dokumen kosong dengan kode status yang ditentukan. Jika nilai diatur ke 201, maka Amazon S3 akan mengembalikan dokumen XML dengan kode status 201. Untuk informasi tentang konten dokumen XML, lihat Objek POST. Jika nilai tidak disetel atau tidak valid, Amazon S3 mengembalikan dokumen kosong dengan kode status 204.  Note Beberapa versi dari Adobe Flash player tidak menangani respons HTTP dengan baik pada bodi kosong. Untuk mendukung pengunggahan melalui Adobe Flash, sebaiknya atur success_action_status menjadi 201.	Tidak

Nama Elemen	Deskripsi	Diperlukan
x-amz-algorithm	Algoritma penandatanganan digunakan untuk mengotentikasi permintaan. Untuk AWS Signature Version 4, nilainya adalah AWS4-HMAC-SHA256 . Bidang ini wajib diisi jika dokumen kebijakan disertakan bersama permintaan.	Diperlukan untuk permintaan yang diautentikasi
x-amz-credential	Selain ID kunci akses Anda, bidang ini juga menyediakan informasi cakupan yang mengidentifikasi wilayah dan layanan yang tanda tangannya valid. Ini harus menjadi cakupan yang sama yang Anda gunakan dalam menghitung kunci penandatanganan untuk perhitungan tanda tangan. Ini adalah string dari bentuk berikut: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Contoh: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</pre> Untuk Amazon S3, string aws-service adalah s3 Untuk daftar aws-region string Amazon S3, lihat Wilayah dan Titik Akhir di Referensi Umum AWS. Ini diperlukan jika dokumen kebijakan disertakan dengan permintaan.	Diperlukan untuk permintaan yang diautentikasi

Nama Elemen	Deskripsi	Diperlukan
x-amz-date	Ini adalah nilai tanggal dalam format ISO8601. Misalnya, 20130728T000000Z . Ini adalah tanggal yang sama yang Anda gunakan dalam membuat kunci penandatanganan (misalnya, 20130728). Nilai ini juga harus sama dengan yang Anda berikan dalam kebijakan (x-amz-date) yang Anda tandatangani. Ini diperlukan jika dokumen kebijakan disertakan dengan permintaan.	Diperlukan untuk permintaan yang diautentikasi
x-amz-security-token	Token keamanan yang digunakan oleh Amazon DevPay dan kredensial sesi Jika permintaan menggunakan Amazon DevPay, diperlukan dua bidang x-amz-security-token formulir: satu untuk token produk dan satu untuk token pengguna. Untuk informasi selengkapnya, lihat Menggunakan DevPay di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Jika permintaan menggunakan kredensial sesi, diperlukan satu x-amz-security-token formulir. Untuk informasi lebih lanjut, lihat Meminta Kredensial Keamanan Sementara dalam Panduan Pengguna IAM.	Tidak

Nama Elemen	Deskripsi	Diperlukan
<code>x-amz-signature</code>	(AWS Signature Version 4) HMAC-SHA256 Hash dari kebijakan keamanan. Bidang ini wajib diisi jika dokumen kebijakan disertakan bersama permintaan.	Diperlukan untuk permintaan yang diautentikasi
<code>x-amz-meta-*</code>	Nama bidang yang dimulai dengan awalan ini adalah metadata yang ditentukan pengguna. Masing-masing disimpan dan dikembalikan sebagai satu set pasangan kunci-nilai. Amazon S3 tidak memvalidasi atau menafsirkan metadata yang ditentukan pengguna. Untuk informasi selengkapnya, lihat PutObject.	Tidak
<code>x-amz-*</code>	Lihat Objek POST (Objek POST) untuk <code>x-amz-*</code> header lainnya.	Tidak
<code>file</code>	Konten file atau teks. File atau konten harus menjadi bidang terakhir dalam formulir. Anda tidak dapat mengunggah lebih dari satu file sekaligus.	Ya

Item bersyarat diperlukan untuk permintaan yang diautentikasi dan opsional untuk permintaan anonim.

Sekarang setelah Anda tahu cara membuat formulir, selanjutnya Anda dapat membuat kebijakan keamanan yang dapat Anda tandatangani. Untuk informasi selengkapnya, lihat [Kebijakan POST](#).

Kebijakan POST

Topik

- [Kedaluwarsa](#)
- [Pencocokan Kondisi](#)
- [Ketentuan](#)
- [Karakter Melarikan Diri](#)

Kebijakan yang diperlukan untuk membuat permintaan yang diautentikasi menggunakan HTTP POST adalah dokumen yang UTF-8 dikodekan base64 yang ditulis dalam JavaScript Object Notation (JSON) yang menentukan kondisi yang harus dipenuhi permintaan. Bergantung pada cara Anda mendesain dokumen kebijakan, Anda dapat mengontrol perincian akses per unggahan, per pengguna, untuk semua unggahan, atau sesuai dengan desain lain yang memenuhi kebutuhan Anda.

Bagian ini menjelaskan kebijakan POST. Misalnya perhitungan tanda tangan menggunakan kebijakan POST, lihat [Contoh: Browser-Based Unggah menggunakan HTTP POST \(Menggunakan AWS Versi Tanda Tangan 4\)](#).

Note

Meskipun dokumen kebijakan bersifat opsional, kami sangat menyarankan Anda menggunakannya untuk mengontrol apa yang diizinkan dalam permintaan. Jika Anda membuat bucket dapat ditulis secara publik, Anda tidak memiliki kendali sama sekali atas pengguna mana yang dapat menulis ke bucket Anda.

Berikut ini adalah contoh dokumen kebijakan POST.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"acl": "public-read" },
    {"bucket": "johnsmith" },
    ["starts-with", "$key", "user/eric/"],
  ]
}
```

Kebijakan POST selalu berisi `conditions` elemen `expiration` dan. Kebijakan contoh menggunakan dua jenis pencocokan kondisi (pencocokan tepat dan mulai dengan pencocokan). Bagian berikut menjelaskan elemen-elemen ini.

Kedaluwarsa

`expiration` Elemen menentukan tanggal kedaluwarsa dan waktu kebijakan POST dalam format tanggal ISO8601 GMT. Misalnya, `2013-08-01T12:00:00.000Z` menentukan bahwa kebijakan POST tidak valid setelah tengah malam GMT pada 1 Agustus 2013.

Pencocokan Kondisi

Berikut adalah tabel yang menjelaskan jenis pencocokan kondisi yang dapat Anda gunakan untuk menentukan kondisi kebijakan POST (dijelaskan di bagian selanjutnya). Meskipun Anda harus menentukan setidaknya satu kondisi untuk setiap bidang formulir yang Anda tentukan dalam formulir, Anda dapat membuat kriteria pencocokan yang lebih kompleks dengan menentukan beberapa kondisi untuk bidang formulir.

Jenis Pencocokan Kondisi	Deskripsi
Kecocokan yang Tepat	Nilai bidang formulir harus sesuai dengan nilai yang ditentukan. Contoh ini menunjukkan bahwa ACL harus diatur ke baca publik: <pre>{"acl": "public-read" }</pre> Contoh ini adalah cara alternatif untuk menunjukkan bahwa ACL harus diatur ke baca publik: <pre>["eq", "\$acl", "public-read"]</pre>
Dimulai Dengan	Nilai harus dimulai dengan nilai yang ditentukan. Contoh ini menunjukkan bahwa kunci objek harus dimulai dengan <code>user/user1</code>:

Jenis Pencocokan Kondisi	Deskripsi
	<pre>["starts-with", "\$key", "user/user1/"]</pre>
Mencocokkan Content-Types dalam Comma-Separated Daftar	Content-Types nilai untuk <code>starts-with</code> kondisi yang menyertakan koma ditafsirkan sebagai daftar. Setiap nilai dalam daftar harus memenuhi syarat agar seluruh kondisi dapat dilewati. Misalnya, dengan kondisi berikut: <pre>["starts-with", "\$Content-Type", "image/"]</pre> Nilai berikut akan melewati kondisi: <pre>"image/jpg,image/png,image/gif"</pre> Nilai berikut tidak akan melewati kondisi: <pre>["image/jpg,text/plain"]  Note Elemen data <code>Content-Type</code> selain diperlakukan sebagai string, terlepas dari keberadaan koma. </pre>
Mencocokkan Konten Apa Pun	Untuk mengonfigurasi kebijakan POST untuk mengizinkan konten apa pun dalam bidang formulir, gunakan <code>starts-with</code> dengan nilai kosong (<code>""</code>). Contoh ini memungkinkan nilai apa pun untuk <code>success_action_redirect</code> : <pre>["starts-with", "\$success_action_redirect", ""]</pre>

Jenis Pencocokan Kondisi	Deskripsi
Menentukan Rentang	Untuk bidang formulir yang menerima rentang, pisahkan batas atas dan bawah dengan koma. Contoh ini memungkinkan ukuran file dari 1 hingga 10 MiB: ["content-length-range", 1048576, 10485760]

Kondisi spesifik yang didukung dalam kebijakan POST dijelaskan di [Ketentuan](#).

Ketentuan

Kebijakan `conditions` dalam POST adalah array objek, yang masing-masing digunakan untuk memvalidasi permintaan. Anda dapat menggunakan kondisi ini untuk membatasi apa yang diizinkan dalam permintaan. Misalnya, kondisi kebijakan sebelumnya memerlukan hal berikut:

- Permintaan harus menentukan nama `johnsmith bucket`.
- Nama kunci objek harus memiliki `user/eric` awalan.
- Objek ACL harus disetel ke `public-read`.

Setiap bidang formulir yang Anda tentukan dalam formulir (kecuali `x-amz-signaturefile,policy,,` dan nama bidang yang memiliki `x-ignore-` awalan) harus muncul dalam daftar kondisi.

Note

Semua variabel dalam formulir diperluas sebelum memvalidasi kebijakan POST. Oleh karena itu, semua pencocokan kondisi harus bertentangan dengan bidang formulir yang diperluas. Misalkan Anda ingin membatasi nama kunci objek Anda ke awalan tertentu (`user/user1`). Dalam hal ini, Anda mengatur bidang formulir kunci ke `user/user1/${filename}`. Kebijakan POST Anda harus `[ "starts-with", "$key", "user/user1/" ]` (jangan masuk `[ "starts-with", "$key", "user/user1/${filename}" ]`). Untuk informasi selengkapnya, lihat [Pencocokan Kondisi](#).

Kondisi dokumen kebijakan dijelaskan dalam tabel berikut.

Nama Elemen	Deskripsi
acl	Menentukan nilai ACL yang harus digunakan dalam pengiriman formulir. Kondisi ini mendukung pencocokan tepat dan jenis kecocokan <code>starts-with</code> kondisi yang dibahas di bagian berikut.
bucket	Menentukan nama bucket yang dapat diterima. Kondisi ini mendukung jenis kecocokan kondisi pencocokan yang tepat.
content-length-range	Ukuran minimum dan maksimum yang diijinkan untuk konten yang diunggah. Kondisi ini mendukung tipe kecocokan <code>content-length-range</code> kecocokan kondisi.
Cache-Control	REST-specific tajuk. Untuk informasi selengkapnya, lihat Objek POST. Kondisi ini mendukung pencocokan yang tepat dan jenis kecocokan <code>starts-with</code> kondisi.
Content-Type	
Content-Disposition	
Content-Encoding	
Expires	
key	Nama kunci yang dapat diterima atau awalan dari objek yang diunggah. Kondisi ini mendukung pencocokan yang tepat dan jenis kecocokan <code>starts-with</code> kondisi.

Nama Elemen	Deskripsi
<code>success_action_redirect</code> <code>redirect</code>	URL yang akan diarahkan klien setelah unggahan berhasil. Kondisi ini mendukung pencocokan yang tepat dan jenis kecocokan <code>starts-with</code> kondisi.
<code>success_action_status</code>	Kode status dikembalikan ke klien setelah berhasil diunggah jika tidak <code>success_action_redirect</code> ditentukan. Kondisi ini mendukung pencocokan yang tepat.
<code>x-amz-algorithm</code>	Algoritma penandatanganan yang harus digunakan selama perhitungan tanda tangan. Untuk AWS Signature Version 4, nilainya adalah <code>AWS4-HMAC-SHA256</code> . Kondisi ini mendukung pencocokan yang tepat.

Nama Elemen	Deskripsi
<code>x-amz-credential</code>	KredenSIAL yang Anda gunakan untuk menghitung tanda tangan. Ini menyediakan ID kunci akses dan informasi ruang lingkup yang mengidentifikasi wilayah dan layanan yang tanda tangannya valid. Ini harus menjadi cakupan yang sama yang Anda gunakan dalam menghitung kunci penandatanganan untuk perhitungan tanda tangan. Ini adalah string dari bentuk berikut: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Contoh: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</pre> Untuk Amazon S3, string <code>aws-service</code> adalah <code>s3</code>. Untuk daftar <code>aws-region</code> string Amazon S3, lihat Wilayah dan Titik Akhir di Referensi Umum AWS. Ini diperlukan jika dokumen kebijakan POST disertakan dengan permintaan. Kondisi ini mendukung pencocokan yang tepat.
<code>x-amz-date</code>	Nilai tanggal yang ditentukan dalam string berformat ISO8601. Misalnya, <code>20130728T000000Z</code>. Tanggal harus sama dengan yang Anda gunakan saat membuat kunci penandatanganan untuk perhitungan tanda tangan. Ini diperlukan jika dokumen kebijakan POST disertakan dengan permintaan. Kondisi ini mendukung pencocokan yang tepat.

Nama Elemen	Deskripsi
<code>x-amz-security-token</code>	Token DevPay keamanan Amazon. Setiap permintaan yang menggunakan Amazon DevPay memerlukan dua bidang <code>x-amz-security-token</code> formulir: satu untuk token produk dan satu untuk token pengguna. Akibatnya, nilai harus dipisahkan dengan koma. Misalnya, jika token pengguna adalah <code>eW91dHVIZQ==</code> dan token produk adalah <code>b0hnNVNKWVJIQTA=</code>, Anda menetapkan entri kebijakan POST ke: { "x-amz-security-token": "eW91dHVIZQ==,b0hnNVNKWVJIQTA=" } . Untuk informasi selengkapnya tentang Amazon DevPay, lihat Menggunakan DevPay di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.
<code>x-amz-meta-*</code>	User-specified metadata. Kondisi ini mendukung pencocokan yang tepat dan jenis kecocokan <code>starts-with</code> kondisi.
<code>x-amz-*</code>	Lihat Objek POST (Objek POST) untuk <code>x-amz-*</code> header lainnya. Kondisi ini mendukung pencocokan yang tepat.

 Note

Jika toolkit menambahkan lebih banyak bidang formulir (misalnya, Flash menambahkan `filename`), Anda harus menambahkannya ke dokumen kebijakan POST. Jika Anda dapat mengontrol fungsi ini, tambahkan awalan `x-ignore-` pada bidang tersebut sehingga Amazon S3 mengabaikan fitur tersebut dan tidak akan memengaruhi versi fitur ini di masa mendatang.

Karakter Melarikan Diri

Karakter yang harus dihilangkan dalam dokumen kebijakan POST dijelaskan dalam tabel berikut.

Urutan Melarikan Diri	Deskripsi
\\	Garis miring terbalik
\\$	Simbol dolar
\b	Spasi Mundur
\f	Umpan formulir
\n	Baris baru
\r	Kembali ke awal baris
\t	Tab Horizontal
\v	Tab vertikal
\ u xxxx	Semua karakter Unicode

Sekarang setelah Anda mengenal formulir dan kebijakan, dan memahami cara kerja penandatanganan, Anda dapat mencoba contoh unggahan POST. Anda perlu menulis kode untuk menghitung tanda tangan. Contoh ini menyediakan formulir sampel, dan kebijakan POST yang dapat Anda gunakan untuk menguji perhitungan tanda tangan Anda. Untuk informasi selengkapnya, lihat [Contoh: Browser-Based Unggah menggunakan HTTP POST \(Menggunakan AWS Versi Tanda Tangan 4\)](#).

Contoh: Browser-Based Unggah menggunakan HTTP POST (Menggunakan AWS Versi Tanda Tangan 4)

Bagian ini menunjukkan contoh penggunaan permintaan HTTP POST untuk mengunggah konten langsung ke Amazon S3.

Untuk informasi selengkapnya tentang Tanda Tangan Versi 4, lihat [Proses Penandatanganan Tanda Tangan Versi 4](#).

Mengunggah File ke Amazon S3 Menggunakan HTTP POST

Contoh ini menyediakan contoh kebijakan POST dan formulir yang dapat Anda gunakan untuk mengunggah file. Topik menggunakan contoh kebijakan dan kredensial fiktif untuk menunjukkan alur kerja dan tanda tangan dan hash kebijakan yang dihasilkan. Anda dapat menggunakan data ini sebagai rangkaian pengujian untuk memverifikasi kode perhitungan tanda tangan Anda.

Contoh menggunakan kredensial contoh berikut perhitungan tanda tangan. Anda dapat menggunakan kredensial ini untuk memverifikasi kode perhitungan tanda tangan Anda. Namun, Anda kemudian harus menggantinya dengan kredensial Anda sendiri saat mengirim permintaan ke AWS.

Parameter	Nilai
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

Contoh Kebijakan dan Formulir

Kebijakan POST berikut mendukung unggahan ke Amazon S3 dengan kondisi tertentu.

```
{ "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" }
]
```

Kebijakan POST ini menetapkan ketentuan berikut pada permintaan:

- Pengunggahan harus dilakukan sebelum tengah hari UTC pada tanggal 30 Desember 2015.
- Konten hanya dapat diunggah ke `sigv4examplebucket`. Bucket harus berada di wilayah yang Anda tentukan dalam cakupan kredensial (parameter `x-amz-credential` formulir), karena tanda tangan yang Anda berikan hanya valid dalam lingkup ini.
- Anda dapat memberikan nama kunci apa pun yang dimulai dengan `user/user1`. Misalnya, `user/user1/MyPhoto.jpg`.
- ACL harus disetel ke `public-read`.
- Jika unggahan berhasil, browser pengguna dialihkan ke `http://sigv4examplebucket.s3.amazonaws.com/successful_upload.html`.
- Objek harus berupa file gambar.
- `x-amz-meta-uuid` harus disetel ke `14365123651274`.
- `x-amz-meta-tag` Bisa berisi nilai apa pun.

Berikut ini adalah Base64-encoded versi kebijakan POST ini. Anda menggunakan nilai ini sebagai perhitungan tanda tangan `StringToSign`.

```
eyJhZXBwaXJhdGlvbiI6ICIyMDE1LTUyMwVDEyOjAwOjAwLjAwMFoiLA0KICAiY29uZG10aW9ucyI6IFsNCiAgICB7ImJ
```

Ketika Anda copy/paste menerapkan kebijakan sebelumnya, itu harus memiliki pengembalian kereta dan baris baru untuk hash yang Anda hitung agar sesuai dengan nilai ini (mis. Teks ASCII, dengan terminator baris CRLF).

Menggunakan contoh kredensial untuk membuat tanda tangan, nilai tanda tangan adalah sebagai berikut (dalam perhitungan tanda tangan, tanggalnya sama dengan `x-amz-date` pada kebijakan (20151229):

```
8afdbf4008c03f22c2cd3cdb72e4afbb1f6a588f3255ac628749a66d7f09699e
```

Formulir contoh berikut menentukan kebijakan POST sebelumnya dan mendukung permintaan POST kesigv4examplebucket. Copy/paste konten dalam editor teks dan simpan sebagai exampleform.html. Anda kemudian dapat mengunggah file gambar ke bucket tertentu menggunakan exampleform.html. Permintaan Anda akan berhasil jika tanda tangan yang Anda berikan cocok dengan tanda tangan yang dihitung Amazon S3.

Note

Anda harus memperbarui nama bucket, tanggal, kredensi, kebijakan, dan tanda tangan dengan nilai yang valid agar berhasil diunggah ke 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>
```

Parameter posting tidak peka huruf besar/kecil. Misalnya, Anda dapat menentukan x-amz-signature atau X-Amz-Signature.

Browser-based mengunggah ke Amazon S3 menggunakan AWS Memperkuat perpustakaan

Bagian ini menjelaskan cara mengunggah file ke Amazon S3 menggunakan per JavaScript pustakaan AWS Amplify.

Untuk informasi tentang pengaturan pustaka AWS Amplify, lihat Instalasi dan Konfigurasi [AWS Amplify](#).

Menggunakan AWS Memperkuat per JavaScript pustakaan untuk Mengunggah File ke Amazon S3

StorageModul perpustakaan AWS Amplify memberikan mekanisme unggahan berbasis browser sederhana untuk mengelola konten pengguna di penyimpanan Amazon S3 publik atau pribadi.

Example: AWS Memperkuat Pengaturan Manual

Contoh berikut menunjukkan pengaturan manual untuk menggunakan modul AWS Amplify Storage. Implementasi default modul Storage menggunakan Amazon S3.

```
import Amplify from 'aws-amplify';
Amplify.configure(
  Auth: {
    identityPoolId: 'XX-XXXX-X:XXXXXXXXX-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: Masukkan data ke Amazon S3

Contoh berikut menunjukkan cara mem put publikasikan data ke Amazon S3.

```
Storage.put('test.txt', 'Hello')
    .then (result => console.log(result))
    .catch(err => console.log(err));
```

Contoh berikut menunjukkan cara merahasi put akan data ke Amazon S3.

```
Storage.put('test.txt', 'Private Content', {
    level: 'private',
    contentType: 'text/plain'
})
    .then (result => console.log(result))
    .catch(err => console.log(err));
```

Untuk informasi selengkapnya tentang menggunakan modul AWS Amplify Storage, lihat [AWS Amplify Storage](#).

Info Selengkapnya

[AWS Memperkuat Mulai Cepat](#)

Header permintaan umum

Tabel berikut menjelaskan header yang dapat digunakan oleh berbagai jenis permintaan REST Amazon S3.

Nama Header	Deskripsi
Authorization	Informasi yang diperlukan untuk otentikasi permintaan. Untuk informasi selengkapnya, buka Header Otentikasi di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon. Untuk permintaan anonim header ini tidak diperlukan.
Access-Control-Request-Method	Daftar metode HTTP yang dikirim sebagai permintaan CORS pra-penerbangan. Jika evaluasi CORS pra-penerbangan berhasil, maka metode yang ditentukan diizinkan untuk digunakan dalam permintaan CORS berikut.
Content-Length	Panjang pesan (tanpa header) sesuai dengan RFC 2616. Header ini diperlukan untuk PUT dan operasi yang memuat XML, seperti logging dan ACL.
Content-Type	Jenis konten sumber daya jika permintaan memiliki konten dalam tubuh. Contoh: text/plain
Content-MD5	Base64 mengkodekan ringkasan pesan MD5 128-bit (tanpa header) menurut RFC 1864. Header ini dapat digunakan sebagai pemeriksaan integritas pesan untuk memverifikasi bahwa data adalah data yang sama dengan yang awalnya dikirim. Meskipun opsional, kami sarankan menggunakan Content-MD5 mekanisme sebagai pemeriksaan integritas ujung ke ujung. Untuk informasi selengkapnya tentang otentikasi permintaan REST, buka Otentikasi REST di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.
Date	Tanggal yang dapat digunakan untuk membuat tanda tangan yang terdapat dalam header Authorization . Jika header

Nama Header	Deskripsi
	Date digunakan untuk penandatanganan, header harus ditentukan dalam format dasar ISO 8601. Dalam hal ini, header <code>x-amz-date</code> tidak diperlukan. Perhatikan bahwa ketika <code>x-amz-date</code> tersedia, nilai header Date akan selalu ditimpa. Jika header Tanggal tidak digunakan untuk penandatanganan, header tersebut bisa menjadi salah satu dari format tanggal penuh yang ditentukan oleh RFC 2616, bagian 3.3. Misalnya, <code>date/time Wed, 01 Mar 2006 12:00:00 GMT</code> adalah date/time header yang valid untuk digunakan dengan Amazon S3. Jika Anda menggunakan header Date untuk penandatanganan, header tersebut harus dalam format <code>YYYYMMDD'T'HHMMSS'Z'</code> dasar ISO 8601. Jika Date ditentukan tetapi tidak dalam format dasar ISO 8601, maka Anda juga harus menyertakan <code>x-amz-date</code> header. Jika Date ditentukan dalam format dasar ISO 8601, ini cukup untuk menandatangani permintaan dan Anda tidak memerlukan header <code>x-amz-date</code>. Untuk informasi selengkapnya, lihat Menangani Tanggal dalam Signature Versi 4 di Glosarium Amazon Web Services.
Expect	Ketika aplikasi Anda menggunakan 100-continue, aplikasi tidak mengirim isi permintaan sampai menerima pengakuan. Jika pesan ditolak berdasarkan header, isi pesan tidak akan terkirim. Header ini hanya dapat digunakan jika Anda mengirim tubuh. Nilai yang Valid: 100-lanjutan

Nama Header	Deskripsi
Host	Untuk permintaan gaya jalur, nilainya adalah <code>s3.amazonaws.com</code>. Untuk permintaan gaya virtual, nilainya adalah <code>BucketName.s3.amazonaws.com</code>. Untuk informasi selengkapnya, buka Virtual Hosting di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Header ini diperlukan untuk HTTP 1.1 (sebagian besar toolkit menambahkan header ini secara otomatis); opsional untuk HTTP/1.0 permintaan.
Origin	Titik akhir yang menentukan nama server dari pemohon awal.
x-amz-content-sha256	Saat menggunakan tanda tangan versi 4 untuk mengotentikasi permintaan, header ini menyediakan hash payload permintaan. Untuk mengetahui informasi selengkapnya, lihat Perhitungan Tanda Tangan untuk Header Otorisasi: Mentransfer Payload dalam Satu Potongan (AWS Versi Tanda Tangan 4). Saat mengunggah objek dalam potongan, Anda menetapkan nilai <code>STREAMING-AWS4-HMAC-SHA256-PAYLOAD</code> untuk menunjukkan bahwa tanda tangan hanya mencakup header dan tidak ada payload. Untuk informasi selengkapnya, lihat Perhitungan Tanda Tangan untuk Header Otorisasi: Mentransfer Payload dalam Beberapa Potongan (Unggahan Terpotong) (AWS Versi Tanda Tangan 4).

Nama Header	Deskripsi
<code>x-amz-date</code>	Tanggal yang digunakan untuk membuat tanda tangan di header Otorisasi. Format harus berupa ISO 8601 dasar dalam format <code>YYYYMMDD'T'HHMMSS'Z'</code> . Misalnya, <code>date/time 20170210T120000Z</code> valid <code>x-amz-date</code> untuk digunakan dengan Amazon S3. <code>x-amz-date</code> bersifat opsional untuk semua permintaan; dapat digunakan untuk mengganti tanggal yang digunakan untuk menandatangani permintaan. Jika Date ditentukan dalam format dasar ISO 8601, <code>x-amz-date</code> tidak diperlukan. Ketika <code>x-amz-date</code> tersedia, nilai header Date akan selalu ditimpa. Untuk informasi selengkapnya, lihat Menangani Tanggal dalam Signature Versi 4 di Glosarium Amazon Web Services.

Nama Header	Deskripsi
<code>x-amz-security-token</code>	Header ini dapat digunakan dalam skenario berikut: • Untuk menyediakan token keamanan untuk DevPay operasi Amazon - Setiap permintaan yang menggunakan Amazon DevPay memerlukan dua <code>x-amz-security-token</code> header: satu untuk token produk dan satu untuk token pengguna. Saat Amazon S3 menerima permintaan yang diautentikasi, Amazon S3 membandingkan tanda tangan yang dihitung dengan tanda tangan yang disediakan. Header multi-nilai yang diformat secara tidak benar yang digunakan untuk menghitung tanda tangan dapat menyebabkan masalah otentikasi. • Untuk memberikan token keamanan saat menggunakan kredensial keamanan sementara - Saat membuat permintaan menggunakan kredensial keamanan sementara yang Anda peroleh dari IAM, Anda harus memberikan token keamanan dengan menggunakan header ini. Untuk mempelajari lebih lanjut tentang kredensial keamanan sementara, lihat Membuat Permintaan. Header ini diperlukan untuk permintaan yang menggunakan Amazon DevPay dan permintaan yang ditandatangani dengan menggunakan kredensial keamanan sementara.

Header respons umum

Tabel berikut menjelaskan header respons yang umum untuk sebagian besar respons Amazon S3.

Nama	Deskripsi
Access-Control-Allow-Credentials	Sebuah Boolean yang menentukan apakah server mengizinkan permintaan CORS berisi kredensial. Jika header <code>Access-Control-Allow-Origin</code> permintaan disetel ke <code>'*'</code> maka header <code>Access-Control-Allow-Credentials</code> respons akan dihilangkan, jika tidak maka akan disetel ke <code>true</code> ketika evaluasi CORS berhasil. Jenis: Boolean Default: Tidak Ada
Access-Control-Allow-Headers	Daftar header HTTP yang diizinkan untuk permintaan CORS Anda. Header <code>Access-Control-Allow-Headers</code> respons dikembalikan untuk evaluasi CORS yang berhasil dan secara eksplisit menentukan semua yang diizinkan <code>Access-Control-Request-Headers</code>. Tipe: String Default: Tidak Ada
Access-Control-Allow-Methods	Daftar yang menentukan metode HTTP mana yang diizinkan. Amazon S3 hanya akan mengizinkan permintaan CORS dari metode CORS yang diizinkan ketika evaluasi CORS berhasil. Tipe: String Default: Tidak Ada
Access-Control-Allow-Origin	Lokasi asal yang diizinkan. Amazon S3 hanya akan mengirimkan header <code>Access-Control-Allow-Origin</code> respons ketika evaluasi CORS berhasil. Jika asal permintaan cocok <code>'*'</code> di asal yang diizinkan konfigurasi CORS maka akan <code>'*'</code> dikembalikan dalam header respons ini alih-alih asal asli.

Nama	Deskripsi
	Tipe: String Default: Tidak Ada
Access-Control-Expose-Headers	Daftar yang memungkinkan server untuk mengidentifikasi header respons yang mengekspos akses untuk aplikasi ketika evaluasi CORS berhasil. Tipe: String Default: Tidak Ada
Access-Control-Max-Age	Waktu dalam detik browser Anda dapat menyimpan respons untuk permintaan pra-penerbangan CORS sebagaimana diidentifikasi oleh sumber daya, metode HTTP, dan asal. Header Access-Control-Max-Age respons hanya dikembalikan ketika evaluasi CORS berhasil. Jenis: Integer Default: Tidak Ada
Vary	Daftar yang menunjukkan header permintaan mana hasil evaluasi CORS bervariasi. Header Vary respons hanya dikembalikan ketika evaluasi CORS berhasil. Tipe: String Default: Tidak Ada
Content-Length	Panjang dalam byte tubuh dalam respons. Tipe: String Default: Tidak Ada
Content-Type	Jenis MIME dari konten. Misalnya, Content-Type: text/html; charset=utf-8 . Tipe: String Default: Tidak Ada

Nama	Deskripsi
Connection	Nilai yang menentukan apakah koneksi ke server terbuka atau tertutup. Jenis: Enum Nilai Valid: open close Default: Tidak Ada
Date	Tanggal dan waktu tanggapan Amazon S3; misalnya, Rabu, 01 Maret 2006 12:00:00 GMT. Tipe: String Default: Tidak Ada
ETag	Tag entitas (ETag) mewakili versi tertentu dari objek. ETag mencerminkan perubahan hanya pada konten suatu objek, bukan metadata-nya. ETag mungkin atau mungkin bukan rangkuman MD5 dari data objek. Apakah itu tergantung pada bagaimana objek dibuat dan bagaimana itu dienkripsi, sebagai berikut: • Objek yang dibuat melalui Konsol Manajemen AWS atau oleh Ob PUT je k,POST Object, atau Copy operasi: • Objek yang berupa plaintext atau dienkripsi dengan enkripsi sisi server dengan kunci terkelola Amazon S3 (SSE-S3) memiliki ETag yang merupakan ringkasan MD5 dari datanya. • Objek yang dienkripsi oleh enkripsi sisi server dengan kunci (SSE-C) atau () kunci AWS Key Management Service (AWS KMS) yang disediakan pelanggan memiliki ET SSE-KMS ag yang bukan merupakan ringkasan MD5 dari data objek mereka. • Objek yang dibuat oleh operasi Multipart Upload atau Upload Part Copy memiliki ETag yang bukan digest MD5, terlepas dari metode enkripsi. Tipe: String

Nama	Deskripsi
Server	Nama server yang membuat respons. Tipe: String Default: AmazonS3
x-amz-delete-marker	Nilai yang menentukan apakah objek yang dikembalikan adalah (<code>true</code>) atau bukan (<code>false</code>) penanda hapus. Jenis: Boolean Nilai Valid: <code>true</code> <code>false</code> Bawaan: salah
x-amz-id-2	Token khusus yang digunakan bersama dengan <code>x-amz-request-id</code> header untuk membantu AWS memecahkan masalah. Untuk informasi tentang Dukungan menggunakan ID permintaan ini, lihat Pemec ahan Masalah Amazon S3 . Tipe: String Default: Tidak Ada
x-amz-request-id	Nilai yang dibuat oleh Amazon S3 yang mengidentifikasi permintaan secara unik. Nilai ini digunakan bersama dengan <code>x-amz-id-2</code> header untuk membantu AWS memecahkan masalah. Untuk informasi tentang Dukungan menggunakan ID permintaan ini, lihat Pemec ahan Masalah Amazon S3 . Tipe: String Default: Tidak Ada
x-amz-server-side-encryption	Algoritma enkripsi sisi server yang digunakan saat menyimpan objek ini di Amazon S3 (misalnya, <code>AES256</code> , <code>aws:kms</code>). Nilai Valid: <code>AES256</code> <code>aws:kms</code>

Nama	Deskripsi
<code>x-amz-version-id</code>	Versi objek. Saat Anda mengaktifkan pembuatan versi, Amazon S3 menghasilkan angka acak untuk objek yang ditambahkan ke bucket. Nilai di UTF-8 kodekan dan URL siap. Ketika Anda PUT sebuah objek dalam bucket di mana pembuatan versi telah ditangguhkan, ID versi selalu <code>null</code> ada. Tipe: String Nilai yang Valid: <code>null</code> apa saja URL-ready, string yang UTF-8 dikodekan Default: <code>null</code>

Tanggapan kesalahan

Bagian ini memberikan informasi referensi tentang kesalahan Amazon S3.

Note

- Secara umum, pemilik bucket S3 ditagih untuk permintaan dengan respons HTTP 200 OK berhasil dan respons kesalahan 4XX klien HTTP. Pemilik bucket tidak ditagih untuk respons kesalahan 5XX server HTTP, seperti 503 Slow Down kesalahan HTTP. Untuk informasi selengkapnya tentang kode kesalahan S3 di bawah HTTP 3XX dan kode 4XX status yang tidak ditagih, lihat [Penagihan untuk respons kesalahan Amazon S3](#) di Panduan Pengguna Amazon S3. Untuk informasi selengkapnya tentang biaya penagihan jika bucket Anda dikonfigurasi sebagai bucket Pembayaran Pemohon, lihat [Cara kerja biaya Pemohon Pembayaran](#) di Panduan Pengguna Amazon S3.
- Dukungan SOAP melalui HTTP tidak digunakan lagi, tetapi SOAP masih tersedia melalui HTTPS. Fitur Amazon S3 baru tidak mendukung SOAP. Alih-alih menggunakan SOAP, sebaiknya gunakan REST API atau AWS SDK.

Topik

- [Respons kesalahan REST](#)
- [Daftar kode kesalahan](#)
- [Daftar Kode Kesalahan Konten Objek SELECT](#)
- [Daftar kode Replication-related kesalahan](#)
- [Daftar kode Tagging-related kesalahan](#)
- [Daftar kode kesalahan terkait anotasi Amazon S3](#)
- [Daftar kode kesalahan Amazon S3 di Outposts](#)
- [Daftar kode kesalahan Amazon S3 Storage Lens](#)
- [Daftar kode kesalahan Lambda Objek Amazon S3](#)
- [Daftar kode kesalahan asinkron Amazon S3](#)
- [Daftar kode kesalahan Amazon S3 Access Grants](#)
- [Daftar kode FSx-related kesalahan Amazon](#)
- [Daftar kode kesalahan Amazon S3 Tables](#)

- [Praktik terbaik kesalahan Amazon S3](#)

Respons kesalahan REST

Ketika terjadi kesalahan, informasi header berisi yang berikut:

- Content-Type: application/xml
- 3xx, 4xx, atau 5xx kode status HTTP yang sesuai

Tubuh respons juga berisi informasi tentang kesalahan. Respon kesalahan sampel berikut menunjukkan struktur elemen respons yang umum untuk semua respons kesalahan 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>
```

Tabel berikut menjelaskan elemen respon kesalahan REST.

Nama	Deskripsi
Code	Kode kesalahan adalah string yang secara unik mengidentifikasi kondisi kesalahan. Ini dimaksudkan untuk dibaca dan dipahami oleh program yang mendeteksi dan menangani kesalahan berdasarkan tipe. Untuk informasi selengkapnya, lihat Daftar kode kesalahan. Tipe: String Leluhur: Error
Error	Wadah untuk semua elemen kesalahan. Tipe: Kontainer Leluhur: Tidak ada

Nama	Deskripsi
Message	Pesan kesalahan berisi deskripsi umum kondisi kesalahan dalam bahasa Inggris. Ini ditujukan untuk audiens manusia. Program sederhana menampilkan pesan secara langsung kepada pengguna akhir jika mereka mengalami kondisi kesalahan yang tidak mereka ketahui atau tidak peduli untuk menanganinya. Program-program canggih dengan penanganan kesalahan yang lebih lengkap dan internasionalisasi yang tepat lebih cenderung mengabaikan pesan kesalahan. Tipe: String Leluhur: Error
RequestId	ID permintaan yang terkait dengan kesalahan. Tipe: String Leluhur: Error
Resource	Bucket atau objek yang terlibat dalam kesalahan. Tipe: String Leluhur: Error

Banyak respons kesalahan berisi data terstruktur tambahan yang dimaksudkan untuk dibaca dan dipahami oleh pengembang yang mendiagnosis kesalahan pemrograman. Misalnya, jika Anda mengirim Content-MD5 header dengan permintaan REST PUT yang tidak cocok dengan ringkasan yang dihitung di server, Anda menerima BadDigest kesalahan. Respons kesalahan juga mencakup sebagai elemen detail ringkasan yang dihitung server, dan intisari yang Anda katakan kepada server untuk diharapkan. Selama pengembangan, Anda dapat menggunakan informasi ini untuk mendiagnosis kesalahan. Dalam produksi, program yang berperilaku baik mungkin menyertakan informasi ini dalam log kesalahannya.

Untuk informasi tentang elemen respons umum, buka [Respons kesalahan](#).

Daftar kode kesalahan

Tabel berikut mencantumkan kode kesalahan Amazon S3.

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
AccessControlListNotSupported	Bucket tidak mengizinkan ACL.	400 Bad Request	Klien
AccessDenied	Akses Ditolak	403 Forbidden	Klien
AccessPointAlreadyOwnedByYou	Titik akses dengan nama yang identik sudah ada di akun Anda.	409 Conflict	Klien
AccountProblem	Ada masalah dengan Akun AWS yang mencegah operasi selesai dengan sukses. Untuk bantuan lebih lanjut, lihat Hubungi Kami .	403 Forbidden	Klien
AllAccessDisabled	Semua akses ke sumber daya Amazon S3 ini telah dinonaktifkan. Untuk bantuan lebih lanjut, lihat Hubungi Kami .	403 Forbidden	Klien
AmbiguousGrantByEmailAddress	Alamat email yang Anda berikan dikaitkan dengan lebih dari satu akun.	400 Bad Request	Klien
AuthorizationHeaderMalformed	Header otorisasi yang Anda berikan tidak valid.	400 Bad Request	N/A

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
AuthorizationQueryParametersError	Parameter kueri otorisasi yang Anda berikan tidak valid.	400 Bad Request	N/A
BadDigest	Nilai checksum Content-MD5 atau yang Anda tentukan tidak cocok dengan apa yang diterima server.	400 Bad Request	Klien
BucketAlreadyExists	Nama bucket yang diminta tidak tersedia. Namespace bucket dibagikan oleh semua pengguna sistem. Tentukan nama yang berbeda dan coba lagi.	409 Conflict	Klien
BucketAlreadyOwnedByYou	Ember yang Anda coba buat sudah ada, dan Anda memilikinya. Amazon S3 mengembalikan kesalahan ini di semua Wilayah AWS kecuali di Wilayah AS Timur (Virginia Utara) (us-east-1). Untuk kompatibilitas lama, jika Anda membuat ulang bucket yang sudah ada yang sudah Anda miliki di us-east-1, Amazon S3 mengembalikan 200 OK dan menyetel ulang daftar kontrol akses bucket (ACL). Untuk Amazon S3 di Outposts, bucket yang Anda coba buat sudah ada di Outpost Anda dan Anda memilikinya.	409 Conflict semua Wilayah kecuali us-east-1)	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
BucketHasAccessPointsAttached	Bucket yang Anda coba hapus memiliki titik akses terlampir. Hapus titik akses Anda sebelum menghapus bucket Anda.	400 Bad Request	Klien
BucketNotEmpty	Bucket yang Anda coba hapus tidak kosong.	409 Conflict	Klien
ClientTokenConflict	Token idempotensi Multi-Region Access Point Anda sudah digunakan untuk permintaan yang berbeda.	409 Conflict	Klien
ConnectionClosedByRequester	Kembali ke penelepon asli ketika terjadi kesalahan saat membaca isi WriteGetObjectResponse .	400 Bad Request	Klien
ConditionalRequestConflict	Operasi yang saling bertentangan terjadi. Jika menggunakan <code>PutObject</code> , Anda dapat mencoba kembali permintaan. Jika menggunakan unggahan multipart , Anda harus memulai <code>CreateMultipartUpload</code> permintaan lain dan mengunggah ulang setiap bagian.	409 Conflict	Klien
CredentialsNotSupported	Permintaan ini tidak mendukung kredensial.	400 Bad Request	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
CrossLocationLoggingProhibited	Cross-Region logging tidak diperbolehkan. Bucket dalam satu Wilayah AWS tidak dapat mencatat informasi ke bucket di Wilayah lain.	403 Forbidden	Klien
DeviceNotActiveError	Perangkat saat ini tidak aktif.	400 Bad Request	Klien
EndpointNotFound	Permintaan langsung ke titik akhir yang benar.	400 Bad Request	Klien
EntityTooSmall	Upload yang Anda usulkan lebih kecil dari ukuran objek minimum yang diizinkan.	400 Bad Request	Klien
EntityTooLarge	Upload yang Anda usulkan melebihi ukuran objek maksimum yang diizinkan. Untuk informasi selengkapnya, lihat titik akhir dan kuota Layanan Penyimpanan Sederhana Amazon di Referensi Umum AWS.	400 Bad Request	Klien
	Unduhan yang Anda usulkan melebihi ukuran maksimum yang diizinkan.	405 Method Not Allowed	Klien
ExpiredToken	Token yang disediakan telah kedaluwarsa.	400 Bad Request	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
<code>IllegalLocationConstraintException</code>	Kesalahan ini mungkin terjadi karena alasan berikut: - Anda mencoba mengakses bucket dari Wilayah yang berbeda dari tempat bucket itu ada. - Anda mencoba membuat bucket dengan batasan lokasi yang sesuai dengan wilayah yang berbeda dari titik akhir regional tempat permintaan dikirim.	400 Bad Request	Klien
<code>IllegalVersioningConfigurationException</code>	Konfigurasi versi yang ditentukan dalam permintaan tidak valid.	400 Bad Request	Klien
<code>IncompleteBody</code>	Anda tidak memberikan jumlah byte yang ditentukan oleh header Content-Length HTTP.	400 Bad Request	Klien
<code>IncorrectEndpoint</code>	Bucket yang ditentukan ada di Wilayah lain. Permintaan langsung ke titik akhir yang benar.	400 Bad Request	Klien
<code>IncorrectNumberOfFilesInPostRequest</code>	POST membutuhkan tepat satu unggahan file per permintaan.	400 Bad Request	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
<code>InlineDataTooLarge</code>	Data sebaris melebihi ukuran maksimum yang diizinkan.	400 Bad Request	Klien
<code>InternalError</code>	Terjadi kesalahan internal. Coba lagi.	500 Internal Server Error	Server
<code>InvalidAccessKeyId</code>	ID kunci AWS akses yang Anda berikan tidak ada dalam catatan kami.	403 Forbidden	Klien
<code>InvalidAccessPoint</code>	Nama atau akun titik akses yang ditentukan tidak valid.	400 Bad Request	Klien
<code>InvalidAccessPointAliasError</code>	Nama alias titik akses yang ditentukan tidak valid.	400 Bad Request	Klien
<code>InvalidAddressingHeader</code>	Anda harus menentukan peran Anonim.	N/A	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
InvalidArgument	Kesalahan ini mungkin terjadi karena alasan berikut: • Per ListBuckets mintaan dibuat ke titik akhir Regional yang berbeda dari Wilayah yang ditentukan dalam bucket-region parameter. • Argumen yang ditentukan tidak valid. • Permintaan tidak memiliki header yang diperlukan. • Argumen yang ditentukan tidak lengkap atau dalam format yang salah. • Argumen yang ditentukan harus memiliki panjang lebih besar dari atau sama dengan 3.	400 Bad Request	Klien
InvalidBucketAclWithObjectOwnership	Bucket tidak dapat memiliki ACL yang disetel dengan BucketOwnerEnforced set ObjectOwnership elan tersebut.	400 Bad Request	Klien
InvalidBucketName	Bucket yang ditentukan tidak valid.	400 Bad Request	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
InvalidBucketOwner AWSAccountID	Nilai parameter pemilik bucket yang diharapkan harus berupa Akun AWS ID.	400 Bad Request	Klien
InvalidBucketState	Permintaan tidak valid untuk status bucket saat ini.	409 Conflict	Klien
InvalidDigest	Nilai checksum Content-MD5 atau yang Anda tentukan tidak valid.	400 Bad Request	Klien
InvalidEncryptionAlgorithmError	Permintaan enkripsi yang Anda tentukan tidak valid. Nilai yang valid adalah AES256.	400 Bad Request	Klien
InvalidHostHeader	Header host yang disediakan dalam permintaan menggunakan pengalamatan gaya yang salah.	400 Bad Request	Klien
InvalidHttpMethod	Permintaan dibuat menggunakan metode HTTP yang tidak terduga.	400 Bad Request	Klien
InvalidLocationConstraint	Batasan lokasi (Wilayah) yang ditentukan tidak valid. Untuk informasi selengkapnya tentang memilih Wilayah untuk bucket Anda, lihat ikhtis ar Bucket .	400 Bad Request	Klien
InvalidObjectState	Operasi ini tidak valid untuk keadaan objek saat ini.	403 Forbidden	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
InvalidPart	Satu atau lebih bagian yang ditentukan tidak dapat ditemukan. Bagian mungkin belum diunggah, atau tag entitas yang ditentukan mungkin tidak cocok dengan tag entitas bagian tersebut.	400 Bad Request	Klien
InvalidPartOrder	Daftar bagian-bagiannya tidak dalam urutan menaik. Daftar bagian harus ditentukan secara berurutan berdasarkan nomor bagian.	400 Bad Request	Klien
InvalidPayer	Semua akses ke objek ini telah dinonaktifkan. Untuk bantuan lebih lanjut, lihat Hubungi Kami .	403 Forbidden	Klien
InvalidPolicyDocument	Isi formulir tidak memenuhi persyaratan yang ditentukan dalam dokumen kebijakan.	400 Bad Request	Klien
InvalidRange	Rentang yang diminta tidak dapat dipenuhi.	416 Rentang yang Diminta Tidak Dapat Dipuaskan	Klien
InvalidRegion	Anda telah mencoba membuat T Multi-Region itik Akses di Wilayah yang belum Anda pilih.	403 Dilarang	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
InvalidRequest	Kesalahan ini mungkin terjadi karena alasan berikut: • Permintaan tanpa halaman ListBuckets dibuat dari akun yang memiliki kuota bucket tujuan umum yang disetujui lebih dari 10.000. Anda harus membuat permintaan berpaginasi untuk mencantumkan bucket di akun dengan lebih dari 10.000 bucket. • Permintaan menggunakan versi tanda tangan yang salah. Gunakan AWS4-HMAC-SHA256 (Tanda Tangan Versi 4). • Titik akses hanya dapat dibuat untuk bucket yang ada. • Titik akses tidak dalam keadaan di mana ia dapat dihapus. • Titik akses hanya dapat dicantumkan untuk bucket yang ada. • Token berikutnya tidak valid. • Setidaknya satu tindakan harus ditentukan dalam aturan siklus hidup.	400 Bad Request	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
	• Setidaknya satu aturan siklus hidup harus ditentukan. • Jumlah aturan siklus hidup tidak boleh melebihi batas 1000 aturan yang diizinkan. • Rentang untuk MaxResults parameter tidak valid. • Permintaan SOAP harus dilakukan melalui koneksi HTTPS. • Akselerasi Transfer Amazon S3 tidak didukung untuk bucket dengan nama yang tidak sesuai dengan DNS. • Akselerasi Transfer Amazon S3 tidak didukung untuk bucket dengan titik (.) dalam namanya. • Titik akhir Amazon S3 Transfer Acceleration hanya mendukung permintaan gaya virtual. • Akselerasi Transfer Amazon S3 tidak dikonfigurasi pada bucket ini. •		

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
	Akselerasi Transfer Amazon S3 dinonaktifkan di bucket ini. - Akselerasi Transfer Amazon S3 tidak didukung di bucket ini. Untuk bantuan, hubungi Dukungan. - Akselerasi Transfer Amazon S3 tidak dapat diaktifkan di bucket ini. Untuk bantuan, hubungi Dukungan. - Nilai yang bertentangan disediakan dalam header HTTP dan parameter kueri. - Nilai yang saling bertentangan disediakan di header HTTP dan kolom formulir POST. - CopyObject permintaan dibuat pada objek yang berukuran lebih besar dari 5GB.		
InvalidSessionException	Dikembalikan jika sesi tidak ada lagi karena waktunya habis atau kedaluwarsa.	400 Bad Request	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
InvalidSignature	Tanda tangan permintaan yang dihitung server tidak cocok dengan tanda tangan yang Anda berikan. Periksa kunci akses AWS rahasia dan metode penandatanganan Anda. Untuk informasi selengkapnya, lihat Men andatangani dan mengotentikasi permintaan REST.	400 Bad Request	Klien
InvalidSecurity	KredenSIAL keamanan yang disediakan tidak valid.	403 Forbidden	Klien
InvalidSOAPRequest	Badan permintaan SOAP tidak valid.	400 Bad Request	Klien
InvalidStorageClass	Kelas penyimpanan yang Anda tentukan tidak valid.	400 Bad Request	Klien
InvalidTargetBucketForLogging	Bucket target untuk logging tidak ada, tidak dimiliki oleh Anda, atau tidak memiliki hibah yang sesuai untuk grup pengiriman log.	400 Bad Request	Klien
InvalidToken	Token yang disediakan cacat atau tidak valid.	400 Bad Request	Klien
InvalidURI	URI yang ditentukan tidak dapat diurai.	400 Bad Request	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
KeyTooLongError	Kuncinya terlalu panjang.	400 Bad Request	Klien
KMS.DisabledException	Permintaan ditolak karena kunci KMS yang ditentukan tidak diaktifkan.	400 Bad Request	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
KMS.InvalidKeyUsageException	Permintaan ditolak karena salah satu alasan berikut: • KeyUsage Nilai kunci KMS tidak kompatibel dengan operasi API. • Algoritma enkripsi atau algoritma penandatanganan yang ditentukan untuk operasi tidak kompatibel dengan jenis materi kunci dalam kunci KMS (KeySpec). Untuk mengenkripsi, mendekripsi, mengenkripsi ulang, dan menghasilkan kunci data, harus ENCRYPT_DECRYPT. KeyUsage Untuk menandatangani dan memverifikasi pesan, KeyUsage harus SIGN_VERIFY. Untuk menghasilkan dan memverifikasi kode otentikasi pesan (MAC), KeyUsage harus GENERATE_VERIFY_MAC. Untuk memperoleh rahasia perjanjian kunci, haruslah KEY_KeyUsage AGREEMENT. Untuk menemukan kunci KMS, gunakan DescribeKey operasi. KeyUsage Untuk menemukan algoritma enkripsi atau penandatanganan yang didukung untuk kunci KMS	400 Bad Request	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
	tertentu, gunakan DescribeKey operasi.		
KMS.KMSInvalidStateException	Permintaan ditolak karena status sumber daya yang ditentukan tidak valid untuk permintaan ini. Pengecualian ini berarti salah satu dari berikut ini: - Status kunci kunci KMS tidak kompatibel dengan operasi. Untuk menemukan status kunci, gunakan DescribeKey operasi. Untuk informasi selengkapnya tentang status kunci mana yang kompatibel dengan setiap operasi KMS, lihat Status kunci AWS KMS keys di Panduan Pengembang AWS Key Management Service. - Untuk operasi kriptografi pada kunci KMS di penyimpanan kunci khusus, pengecualian ini mewakili kegagalan umum dengan banyak kemungkinan penyebab. Untuk mengidentifikasi penyebabnya, lihat pesan kesalahan yang menyertainya pengecualian.	400 Bad Request	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
<code>KMS.NotFoundException</code>	Permintaan ditolak karena entitas atau sumber daya yang ditentukan tidak dapat ditemukan.	400 Bad Request	Klien
<code>MalformedACLError</code>	ACL yang Anda berikan tidak terbentuk dengan baik atau tidak divalidasi terhadap skema yang kami publikasikan.	400 Bad Request	Klien
<code>MalformedPOSTRequest</code>	Tubuh permintaan POST Anda bukanlah multipart/form-data yang dibentuk dengan baik.	400 Bad Request	Klien
<code>MalformedXML</code>	XML yang Anda berikan tidak terbentuk dengan baik atau tidak divalidasi terhadap skema yang kami publikasikan.	400 Bad Request	Klien
<code>MaxMessageLengthExceeded</code>	Permintaan Anda terlalu besar.	400 Bad Request	Klien
<code>MaxPostPreDataLengthExceededError</code>	Bidang permintaan POST Anda sebelum file upload terlalu besar.	400 Bad Request	Klien
<code>MetadataTooLarge</code>	Header metadata Anda melebihi ukuran metadata maksimum yang diizinkan.	400 Bad Request	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
MethodNotAllowed	Metode yang ditentukan tidak diperbolehkan terhadap sumber daya ini.	405 Method Not Allowed	Klien
MissingAttachment	Lampiran SOAP diharapkan, tetapi tidak ada yang ditemukan.	400 Permintaan Buruk	Klien
MissingAuthenticationToken	Permintaan itu tidak ditandatangani.	403 Dilarang	Klien
MissingContentLength	Anda harus memberikan header Content-Length HTTP.	411 Length Required	Klien
MissingRequestBodyError	Anda mengirim dokumen XML kosong sebagai permintaan.	400 Bad Request	Klien
MissingSecurityElement	Permintaan SOAP 1.1 tidak memiliki elemen keamanan.	400 Bad Request	Klien
MissingSecurityHeader	Permintaan Anda tidak memiliki header yang diperlukan.	400 Bad Request	Klien
NoLoggingStatusForKey	Tidak ada yang namanya subresource status logging untuk kunci.	400 Bad Request	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
NoSuchAsyncRequest	Permintaan yang ditentukan tidak ditemukan.	404 Not Found	Klien
NoSuchBucket	Bucket yang ditentukan tidak ada.	404 Not Found	Klien
NoSuchBucketPolicy	Bucket yang ditentukan tidak memiliki kebijakan bucket.	404 Not Found	Klien
NoSuchCORSConfiguration	Bucket yang ditentukan tidak memiliki konfigurasi CORS.	404 Not Found	Klien
NoSuchKey	Kunci yang ditentukan tidak ada.	404 Not Found	Klien
NoSuchLifecycleConfiguration	Konfigurasi siklus hidup yang ditentukan tidak ada.	404 Not Found	Klien
NoSuchMultiRegionAccessPoint	Titik Multi-Region Akses yang ditentukan tidak ada.	404 Not Found	Klien
NoSuchObjectLockConfiguration	Objek yang ditentukan tidak memiliki ObjectLock konfigurasi.	404 Not Found	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
NoSuchWebsiteConfiguration	Bucket yang ditentukan tidak memiliki konfigurasi situs web.	404 Not Found	Klien
NoSuchTagSet	Tag yang ditentukan tidak ada.	404 Not Found	Klien
NoSuchUpload	Unggahan multipart yang ditentukan tidak ada. ID unggahan mungkin tidak valid, atau unggahan multipart mungkin telah dibatalkan atau diselesaikan.	404 Not Found	Klien
NoSuchVersion	ID versi yang ditentukan dalam permintaan tidak cocok dengan versi yang ada.	404 Not Found	Klien
NotDeviceOwnerError	Perangkat yang menghasilkan token tidak dimiliki oleh pengguna yang diautentikasi.	400 Bad Request	Klien
NotImplemented	Header yang Anda berikan menyiratkan fungsionalitas yang tidak diimplementasikan.	501 Not Implemented	Server
NotModified	Sumber daya tidak berubah.	304 Not Modified	Server

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
NoTransformationDefined	Tidak ada transformasi yang ditemukan untuk Titik Akses Lambda Objek ini.	404 Not Found	Klien
NotSignedUp	Akun Anda tidak terdaftar untuk layanan Amazon S3. Anda harus mendaftar sebelum dapat menggunakan Amazon S3. Anda dapat mendaftar di URL berikut: https://aws.amazon.com/s3	403 Forbidden	Klien
ObjectLockConfigurationNotFoundError	Konfigurasi Object Lock tidak ada untuk bucket ini.	404 Not Found	Klien
OwnershipControlsNotFoundError	Kontrol kepemilikan bucket tidak ditemukan.	404 Not Found	Klien
OperationAborted	Operasi bersyarat yang saling bertentangan saat ini sedang berlangsung terhadap sumber daya ini. Coba lagi.	409 Conflict	Klien
PermanentRedirect	Bucket yang Anda coba akses harus ditangani menggunakan titik akhir yang ditentukan. Kirim semua permintaan masa depan ke titik akhir ini.	301 Moved Permanently	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
PermanentRedirectControlError	Operasi API yang Anda coba akses harus ditangani menggunakan titik akhir yang ditentukan. Kirim semua permintaan masa depan ke titik akhir ini.	301 Moved Permanently	Klien
PreconditionFailed	Setidaknya salah satu prasyarat yang Anda tentukan tidak berlaku.	412 Precondition Failed	Klien
Redirect	Pengalihan sementara. Anda diarahkan ke bucket saat server Domain Name System (DNS) sedang diperbarui.	307 Temporary Redirect	Klien
RequestHeaderSectionTooLarge	Parameter header permintaan dan kueri yang digunakan untuk membuat permintaan melebihi ukuran maksimum yang diizinkan.	400 Bad Request	Klien
RequestIsNotMultipartContent	Permintaan bucket POST harus dari enclosure-type multipart/form-data.	412 Precondition Failed	Klien
RequestTimeout	Koneksi soket Anda ke server tidak dibaca dari atau ditulis dalam periode batas waktu.	400 Bad Request	Klien
RequestTimeTooSkewed	Perbedaan antara waktu permintaan dan waktu server terlalu besar.	403 Forbidden	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
RequestTorrentOfBucketError	Meminta file torrent dari bucket tidak diizinkan.	400 Bad Request	Klien
ResponseInterrupted	Kembali ke penelepon asli ketika terjadi kesalahan saat membaca isi WriteGetObjectResponse .	400 Bad Request	Klien
RestoreAlreadyInProgress	Pemulihan objek sudah berlangsung.	409 Conflict	Klien
ServerSideEncryptionConfigurationNotFoundError	Konfigurasi enkripsi sisi server tidak ditemukan.	400 Bad Request	Klien
ServiceUnavailable	Layanan tidak dapat menangani permintaan.	503 Service Unavailable	Server
SignatureDoesNotMatch	Tanda tangan permintaan yang dihitung server tidak cocok dengan tanda tangan yang Anda berikan. Periksa kunci akses AWS rahasia dan metode penandatanganan Anda. Untuk informasi selengkapnya, lihat Otentikasi REST dan Otentikasi SO AP .	403 Forbidden	Klien
SlowDown	Harap kurangi tarif permintaan Anda.	503 Slow Down	Server

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
503 SlowDown	Perlambat	503 Slow Down	Server
TagPolicyException	Kebijakan tag tidak mengizinkan nilai yang ditentukan untuk kunci tag berikut.	400 Bad Request	Klien
TemporaryRedirect	Anda diarahkan ke bucket saat server Domain Name System (DNS) sedang diperbarui.	307 Temporary Redirect	Klien
TokenCodeInvalidError	Kode and/or token nomor seri yang Anda berikan tidak valid.	400 Bad Request	Klien
TokenRefreshRequired	Token yang disediakan harus diperbarui.	400 Bad Request	Klien
TooManyAccessPoints	Anda telah mencoba membuat lebih banyak titik akses daripada yang diizinkan untuk akun. Untuk informasi selengkapnya, lihat titik akhir dan kuota Layanan Penyimpanan Sederhana Amazon di Referensi Umum AWS.	400 Bad Request	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
TooManyBuckets	Anda telah mencoba membuat lebih banyak bucket daripada yang diizinkan untuk akun. Untuk informasi selengkapnya, lihat titik akhir dan kuota Layanan Penyimpanan Sederhana Amazon di Referensi Umum AWS.	400 Bad Request	Klien
TooManyMultiRegionAccessPointregionsError	Anda telah mencoba membuat Titik Multi-Region Akses dengan lebih banyak Wilayah daripada yang diizinkan untuk akun. Untuk informasi selengkapnya, lihat titik akhir dan kuota Layanan Penyimpanan Sederhana Amazon di Referensi Umum AWS.	400 Bad Request	Klien
TooManyMultiRegionAccessPoints	Anda telah mencoba membuat lebih banyak Po Multi-Region in Akses daripada yang diizinkan untuk akun. Untuk informasi selengkapnya, lihat titik akhir dan kuota Layanan Penyimpanan Sederhana Amazon di Referensi Umum AWS.	400 Bad Request	Klien
UnauthorizedAccessError	Hanya berlaku di Wilayah China. Dikembalikan ketika permintaan dibuat ke bucket yang tidak memiliki lisensi ICP. Untuk informasi selengkapnya, lihat ICP Record al.	403 Forbidden	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
UnexpectedContent	Permintaan ini berisi konten yang tidak didukung.	400 Bad Request	Klien
UnexpectedIPError	Hanya berlaku di Wilayah China. Permintaan ini ditolak karena IP tidak terduga.	403 Forbidden	Klien
UnsupportedArgument	Permintaan berisi argumen yang tidak didukung.	400 Bad Request	Klien
UnsupportedSignature	Permintaan yang diberikan ditandatangani dengan versi Token STS yang tidak didukung atau versi tanda tangan tidak didukung.	400 Bad Request	Klien
UnresolvableGrantByEmailAddress	Alamat email yang Anda berikan tidak cocok dengan akun apa pun yang tercatat.	400 Bad Request	Klien
UserKeyMustBeSpecified	Permintaan bucket POST harus berisi nama bidang yang ditentukan. Jika ditentukan, periksa urutan bidang.	400 Bad Request	Klien
NoSuchAccessPoint	Titik akses yang ditentukan tidak ada.	404 Not Found	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
InvalidTag	Permintaan Anda berisi input tag yang tidak valid. Misalnya, permintaan Anda mungkin berisi kunci duplikat, kunci atau nilai yang terlalu panjang, atau tag sistem.	400 Bad Request	Klien
MalformedPolicy	Polis Anda berisi prinsip yang tidak valid.	400 Bad Request	Klien

Daftar Kode Kesalahan Konten Objek SELECT

Important

Amazon S3 Select tidak lagi tersedia untuk pelanggan baru. Pelanggan Amazon S3 Select yang sudah ada dapat terus menggunakan fitur ini seperti biasa. [Pelajari selengkapnya](#)

Tabel berikut berisi kesalahan khusus yang SELECT Object Content mungkin kembali. Untuk informasi umum tentang kesalahan Amazon S3 dan daftar kode kesalahan, lihat [Tanggapan kesalahan](#).

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
AmbiguousFieldName	Nama bidang cocok dengan beberapa bidang dalam file. Periksa ekspresi SQL dan file, dan coba lagi.	400	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
Busy	Layanan tidak tersedia. Coba lagi nanti.	503	Klien
CastFailed	Upaya untuk mengkonversi dari satu tipe data ke yang lain menggunakan CAST gagal dalam ekspresi SQL.	400	Klien
ColumnTooLong	Panjang kolom dalam hasilnya lebih besar maxCharsPerColumn dari 1 MB.	400	Klien
CSVEscapingRecordDelimiter	Pembatas catatan yang dikutip ditemukan dalam file. Untuk mengizinkan pembatas rekaman yang dikutip, set AllowQuotedRecordDelimiter ke 'TRUE'	400	Klien
CSVParsingError	Terjadi kesalahan saat mengurai file CSV. Periksa file dan coba lagi.	400	Klien
CSVUnescapedQuote	Kutipan yang tidak terlupakan ditemukan saat mengurai file CSV. Untuk mengizinkan pembatas rekaman yang dikutip, set AllowQuotedRecordDelimiter ke 'TRUE'	400	Klien
EmptyRequestBody	Badan permintaan tidak boleh kosong.	400	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
EvaluatorBindingDoesNotExist	Nama kolom atau jalur yang disediakan tidak ada dalam ekspresi SQL.	400	Klien
EvaluatorInvalidArguments	Ada jumlah argumen yang salah dalam panggilan fungsi dalam ekspresi SQL.	400	Klien
EvaluatorInvalidTimestampFormatPattern	String format stempel waktu dalam ekspresi SQL tidak valid.	400	Klien
EvaluatorInvalidTimestampFormatPatternSymbol	Pola format stempel waktu berisi simbol dalam ekspresi SQL yang tidak valid.	400	Klien
EvaluatorInvalidTimestampFormatPatternSymbolForParsing	Pola format stempel waktu berisi simbol format yang valid yang tidak dapat diterapkan pada penguraian stempel waktu dalam ekspresi SQL.	400	Klien
EvaluatorInvalidTimestampFormatPatternToken	Pola format stempel waktu berisi token dalam ekspresi SQL yang tidak valid.	400	Klien
EvaluatorLikePatternInvalidEscapeSequence	Argumen yang diberikan pada LIKE ekspresi itu tidak valid.	400	Klien
EvaluatorNegativeLimit	LIMIT tidak boleh negatif.	400	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
EvaluatorTimestampFormatPatternDuplicateFields	Pola format stempel waktu berisi beberapa penentu format yang mewakili bidang stempel waktu dalam ekspresi SQL.	400	Klien
EvaluatorTimestampFormatPatternHourClockAmPmMismatch	Pola format stempel waktu berisi simbol format jam 12 jam sehari tetapi tidak juga berisi AM/PM bidang, atau berisi penentu format jam sehari 24 jam dan berisi AM/PM bidang dalam ekspresi SQL.	400	Klien
EvaluatorUnterminatedTimestampFormatPatternToken	Pola format stempel waktu berisi token yang tidak diakhiri dalam ekspresi SQL.	400	Klien
ExpressionTooLong	Ekspresi SQL terlalu panjang. Panjang byte maksimum untuk ekspresi SQL adalah 256 KB.	400	Klien
ExternalEvalException	Kueri tidak dapat dievaluasi. Periksa file dan coba lagi.	400	Klien
IllegalSqlFunctionArgument	Argumen ilegal digunakan dalam fungsi SQL.	400	Klien
IncorrectSqlFunctionArgumentType	Jenis argumen yang salah ditentukan dalam panggilan fungsi dalam ekspresi SQL.	400	Klien
IntegerOverflow	Overflow atau underflow integer terjadi dalam ekspresi SQL.	400	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
<code>InternalError</code>	Terjadi kesalahan internal.	500	Klien
<code>InvalidCast</code>	Upaya untuk mengkonversi dari satu tipe data ke yang lain menggunakan CAST gagal dalam ekspresi SQL.	400	Klien
<code>InvalidColumnIndex</code>	Indeks kolom dalam ekspresi SQL tidak valid.	400	Klien
<code>InvalidCompressionFormat</code>	File tidak dalam format kompresi yang didukung. Hanya GZIP dan BZIP2 yang didukung.	400	Klien
<code>InvalidDataSource</code>	Jenis sumber data tidak valid. Hanya CSV, JSON, dan Parquet yang didukung.	400	Klien
<code>InvalidDataType</code>	Ekspresi SQL berisi tipe data yang tidak valid.	400	Klien
<code>InvalidExpressionType</code>	<code>ExpressionType</code> Nilai tidak valid. Hanya ekspresi SQL yang didukung.	400	Klien
<code>InvalidFileHeaderInfo</code>	<code>FileHeaderInfo</code> Nilai tidak valid. Hanya <code>NONE</code> , <code>USE</code> , dan <code>IGNORE</code> didukung.	400	Klien
<code>InvalidJsonType</code>	<code>JsonType</code> Nilai tidak valid. Hanya <code>DOCUMENT</code> dan <code>LINES</code> yang didukung.	400	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
InvalidKeyPath	Jalur kunci dalam ekspresi SQL tidak valid.	400	Klien
InvalidQuoteFields	QuoteFields Nilai tidak valid. Hanya ALWAYS dan ASNEEDED yang didukung.	400	Klien
InvalidRequestParameter	Nilai parameter dalam SelectRequest elemen tidak valid. Periksa dokumentasi API layanan dan coba lagi.	400	Klien
InvalidScanRange	Rentang pemindaian yang disediakan tidak valid.	400	Klien
InvalidTableAlias	Ekspresi SQL berisi alias tabel yang tidak valid.	400	Klien
InvalidTextEncoding	Jenis pengkodean tidak valid. Hanya UTF-8 pengkodean yang didukung.	400	Klien
JSONParsingError	Terjadi kesalahan saat mengurai file JSON. Periksa file dan coba lagi.	400	Klien
LexerInvalidChar	Ekspresi SQL berisi karakter yang tidak valid.	400	Klien
LexerInvalidIONLiteral	Ekspresi SQL berisi operator yang tidak valid.	400	Klien
LexerInvalidLiteral	Ekspresi SQL berisi operator yang tidak valid.	400	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
LexerInvalidOperator	Ekspresi SQL berisi literal yang tidak valid.	400	Klien
LikeInvalidInputs	Argumen yang diberikan pada LIKE klausa dalam ekspresi SQL tidak valid.	400	Klien
MalformedXML	XML yang disediakan tidak terbentuk dengan baik atau tidak divalidasi terhadap skema yang kami publikasikan. Periksa dokumentasi layanan dan coba lagi.	400	Klien
MaxOperatorsExceeded	Gagal mengurai ekspresi SQL, coba kurangi kompleksitas. Misalnya, kurangi jumlah operator yang digunakan.	400	Klien
MethodNotAllowed	Metode yang ditentukan tidak diperbolehkan terhadap sumber daya ini.	405 Method Not Allowed	Klien
MissingRequiredParameter	En SelectRequest titas kehilangan parameter yang diperlukan. Periksa dokumentasi layanan dan coba lagi.	400	Klien
MultipleDataSourcesUnsupported	Beberapa sumber data tidak didukung.	400	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
NumberFormatError	Terjadi kesalahan saat mengurai angka. Kesalahan ini dapat disebabkan oleh luapan atau luapan bilangan bulat.	400	Klien
ObjectSerializationConflict	InputSerialization menentukan lebih dari satu format (CSV, JSON, atau Parquet), atau OutputSerialization menentukan lebih dari satu format (CSV atau JSON). Untuk InputSerialization dan OutputSerialization, Anda hanya dapat menentukan satu format untuk masing-masing.	400	Klien
OverMaxColumn	Jumlah kolom dalam hasil lebih besar dari jumlah kolom maksimum yang diijinkan.	400	Klien
OverMaxParquetBlockSize	File Parquet berada di atas ukuran grup baris maksimum.	400	Klien
OverMaxRecordSize	Panjang catatan dalam input atau hasil lebih besar dari maxCharsPerRecord batas 1 MB.	400	Klien
ParquetParsingError	Terjadi kesalahan saat mengurai file Parquet. Periksa file dan coba lagi.	400	Klien
ParquetUnsupportedCompressionCodec	Codec kompresi parket yang ditentukan tidak didukung.	400	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
ParseAsteriskIsNotAloneInSelectList	Ekspresi lain tidak diperbolehkan dalam SELECT daftar ketika * digunakan tanpa notasi titik dalam ekspresi SQL.	400	Klien
ParseCannotMixSqbandWildcardInSelectList	Tidak dapat [] mencampur dan * dalam ekspresi yang sama dalam SELECT daftar dalam ekspresi SQL.	400	Klien
ParseCastArity	Ekspresi SQL CAST memiliki aritas yang salah.	400	Klien
ParseEmptySelect	Ekspresi SQL berisi SELECT klausa kosong.	400	Klien
ParseExpected2TokenTypes	Token yang diharapkan dalam ekspresi SQL tidak ditemukan.	400	Klien
ParseExpectedArgumentDelimiter	Pembatas argumen yang diharapkan dalam ekspresi SQL tidak ditemukan.	400	Klien
ParseExpectedDatePart	Bagian tanggal yang diharapkan dalam ekspresi SQL tidak ditemukan.	400	Klien
ParseExpectedExpression	Ekspresi SQL yang diharapkan tidak ditemukan.	400	Klien
ParseExpectedIdentifierForAlias	Pengidentifikasi yang diharapkan untuk alias dalam ekspresi SQL tidak ditemukan.	400	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
ParseExpectedIdentForAt	Pengidentifikasi yang diharapkan untuk AT nama dalam ekspresi SQL tidak ditemukan.	400	Klien
ParseExpectedIdentForGroupName	GROUP tidak didukung dalam ekspresi SQL.	400	Klien
ParseExpectedKeyword	Kata kunci yang diharapkan dalam ekspresi SQL tidak ditemukan.	400	Klien
ParseExpectedLeftParenAfterCast	Kurung kiri yang diharapkan setelah CAST dalam ekspresi SQL tidak ditemukan.	400	Klien
ParseExpectedLeftParenBuiltinFunctionCall	Kurung kiri yang diharapkan dalam ekspresi SQL tidak ditemukan.	400	Klien
ParseExpectedLeftParenValueConstructor	Kurung kiri yang diharapkan dalam ekspresi SQL tidak ditemukan.	400	Klien
ParseExpectedMember	Ekspresi SQL berisi penggunaan yang tidak didukung. MEMBER	400	Klien
ParseExpectedNumber	Angka yang diharapkan dalam ekspresi SQL tidak ditemukan.	400	Klien
ParseExpectedRightParenBuiltinFunctionCall	Karakter tanda kurung kanan yang diharapkan dalam ekspresi SQL tidak ditemukan.	400	Klien
ParseExpectedTokenType	Token yang diharapkan dalam ekspresi SQL tidak ditemukan.	400	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
ParseExpectedTypeName	Nama tipe yang diharapkan dalam ekspresi SQL tidak ditemukan.	400	Klien
ParseExpectedWhenClause	Klausula WHEN yang diharapkan dalam ekspresi SQL tidak ditemukan. CASE tidak didukung.	400	Klien
ParseInvalidContextForWildcardInSelectList	Penggunaan * dalam SELECT daftar dalam ekspresi SQL tidak valid.	400	Klien
ParseInvalidPathComponent	Ekspresi SQL berisi komponen jalur yang tidak valid.	400	Klien
ParseInvalidTypeParameter	Ekspresi SQL berisi nilai parameter yang tidak valid.	400	Klien
ParseMalformedJoin	JOIN tidak didukung dalam ekspresi SQL.	400	Klien
ParseMissingIdentifierAt	Pengidentifikasi yang diharapkan setelah simbol @ dalam ekspresi SQL tidak ditemukan.	400	Klien
ParseNonUnaryAggregateFunctionCall	Hanya satu argumen yang didukung untuk fungsi agregat dalam ekspresi SQL.	400	Klien
ParseSelectMissingFrom	Ekspresi SQL berisi yang hilang FROM setelah SELECT daftar.	400	Klien
ParseUnexpectedKeyword	Ekspresi SQL berisi kata kunci yang tidak terduga.	400	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
ParseUnexpectedOperator	Ekspresi SQL berisi operator tak terduga.	400	Klien
ParseUnexpectedTerm	Ekspresi SQL berisi istilah yang tidak terduga.	400	Klien
ParseUnexpectedToken	Ekspresi SQL berisi token tak terduga.	400	Klien
ParseUnknownOperator	Ekspresi SQL berisi operator yang tidak valid.	400	Klien
ParseUnsupportedAlias	Ekspresi SQL berisi penggunaan yang tidak didukung. ALIAS	400	Klien
ParseUnsupportedCallWithStar	Hanya COUNT dengan (*) sebagai parameter yang didukung dalam ekspresi SQL.	400	Klien
ParseUnsupportedCase	Ekspresi SQL berisi penggunaan yang tidak didukung. CASE	400	Klien
ParseUnsupportedCaseClause	Ekspresi SQL berisi penggunaan yang tidak didukung. CASE	400	Klien
ParseUnsupportedLiteralsGroupBy	Ekspresi SQL berisi penggunaan yang tidak didukung. GROUP BY	400	Klien
ParseUnsupportedSelect	Ekspresi SQL berisi penggunaan yang tidak didukung. SELECT	400	Klien
ParseUnsupportedSyntax	Ekspresi SQL berisi sintaks yang tidak didukung.	400	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
ParseUnsupportedToken	Ekspresi SQL berisi token yang tidak didukung.	400	Klien
TruncatedInput	Dekompresi objek gagal. Periksa apakah objek dikompresi dengan benar menggunakan format yang ditentukan dalam permintaan.	400	Klien
UnauthorizedAccess	Anda tidak berwenang untuk melakukan operasi ini.	401	Klien
UnrecognizedFormatException	Jenis rekaman tidak valid.	400	Klien
UnsupportedFunction	S3 menemukan fungsi SQL yang tidak didukung.	400	Klien
UnsupportedParquetType	Jenis Parquet yang ditentukan tidak didukung.	400	Klien
UnsupportedRangeHeader	Header rentang tidak didukung untuk operasi ini.	400	Klien
UnsupportedScanRangeInput	Kueri rentang pemindaian tidak didukung pada jenis objek ini.	400	Klien
UnsupportedSqlOperation	S3 mengalami operasi SQL yang tidak didukung.	400	Klien
UnsupportedSqlStructure	S3 menemukan struktur SQL yang tidak didukung. Periksa Referensi SQL.	400	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
UnsupportedStorageClass	Kelas penyimpanan tidak didukung. Hanya STANDARD, STANDARD_IA, dan kelas ONEZONE_IA penyimpanan didukung.	400	Klien
UnsupportedSyntax	Sintaksnya tidak valid.	400	Klien
UnsupportedTypeForQuerying	Kueri Anda berisi tipe yang tidak didukung untuk perbandingan (misalnya memverifikasi bahwa jenis kolom Parquet INT96 lebih besar dari 0).	400	Klien
ValueParseFailure	Kegagalan penguraian stempel waktu terjadi dalam ekspresi SQL.	400	Klien

Daftar kode Replication-related kesalahan

Tabel berikut berisi kesalahan khusus yang mungkin dikembalikan oleh Replication operasi. Untuk informasi umum tentang kesalahan Amazon S3 dan daftar kode kesalahan, lihat [Tanggapan kesalahan](#).

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
InvalidArgument	Kesalahan ini mungkin terjadi karena alasan berikut:	400	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
	<Account> Elemen kosong. Itu harus berisi ID akun yang valid. • Yang Akun AWS ditentukan dalam <Account> elemen harus cocok dengan pemilik bucket tujuan. • ReplicationTime-Status harus mengandung nilai. • ReplicationTime-ReplicationTimeValue harus mengandung nilai. • Replication-ReplicationTimeValue-Minutes nilainya harus 15. • ReplicationMetrics harus berisi aStatus. • ReplicationMetrics harus berisi EventThreshold . • EventThreshold-ReplicationTimeValue-Minutes nilainya harus 15. •		

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
	Rule ID tidak boleh mengandung karakter non-ASCII.		
InvalidRequest	Kesalahan ini mungkin terjadi karena alasan berikut: • <code><Owner></code> In <code><AccessControlTranslation></code> memiliki nilai, jadi <code><Account></code> elemen harus ditentukan. • <code><Account></code> Elemen kosong. Itu harus berisi ID akun yang valid. • Replikasi destination harus berisi keduanya <code>ReplicationTime</code> dan <code>Metrics</code>, atau tidak keduanya. • <code>ReplicationTime</code> dan <code>ReplicationMetrics</code> harus memiliki status yang sama. • Kontrol Waktu Replikasi S3 (S3 RTC) tidak didukung dalam hal ini Wilayah AWS.	400	Klien

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
ReplicationConfigurationNotFoundError	Tidak ada konfigurasi replikasi untuk bucket ini.	404 Tidak Ditemukan	Klien

Daftar kode Tagging-related kesalahan

Tabel berikut berisi kesalahan khusus yang mungkin dikembalikan oleh `ListTagsForResource` operasi `TagResource` `UntagResource`, dan untuk grup Lensa Penyimpanan. Untuk informasi umum tentang kesalahan Amazon S3 umum dan daftar kode kesalahan, lihat [Tanggapan kesalahan](#).

Kode Kesalahan	Deskripsi	Kode Status HTTP	Awalan Kode Kesalahan SOAP
InvalidRequest	ARN Wilayah AWS dalam sumber daya tidak cocok dengan Wilayah yang ditentukan dalam permintaan ini. AWS Akun di sumber daya ARN tidak cocok dengan ID akun yang ditentukan dalam permintaan ini. AWS Partisi di ResourceLearn tidak valid.	400 Bad Request	Tidak didukung
InvalidTag	Permintaan ini berisi kunci tag atau nilai yang tidak valid. Karakter yang valid meliputi yang berikut: <code>[a-zA-Z+-=._:/]</code> . Kunci tag dapat berisi hingga 128 karakter. Nilai tag dapat berisi hingga 256 karakter. Ada kunci tag	400 Bad Request	Tidak didukung

Kode Kesalahan	Deskripsi	Kode Status HTTP	Awalan Kode Kesalahan SOAP
	duplikat dalam permintaan Anda. User-defined kunci tag tidak bisa dimulai denganaws : .		
NoSuchResource	Sumber daya yang ditentukan tidak ada.	404 Not Found	Tidak didukung
TagPolicyException	Kebijakan tag tidak mengizinkan nilai yang ditentukan untuk kunci tag berikut.	400 Bad Request	Tidak didukung
TooManyTags	Jumlah tag melebihi batas 50 tag.	400 Bad Request	Tidak didukung

Daftar kode kesalahan terkait anotasi Amazon S3

Tabel berikut berisi kesalahan khusus yang mungkin dikembalikan oleh operasi anotasi (PutObjectAnnotationGetObjectAnnotation,ListObjectAnnotations,DeleteObjectAnnotation dan CopyObject perilaku terkait anotasi. Untuk informasi umum tentang kesalahan Amazon S3 dan daftar kode kesalahan, lihat [Tanggapan kesalahan](#).

Kode Kesalahan	Deskripsi	Kode Status HTTP	Awalan Kode Kesalahan SOAP
AccessDenied	Akses Ditolak. Pemanggil tidak memiliki izin untuk melakukan operasi anotasi yang diminta, atau objek dilindungi oleh Object Lock dalam mode COMPLIANCE.	403 Forbidden	Tidak didukung

Kode Kesalahan	Deskripsi	Kode Status HTTP	Awalan Kode Kesalahan SOAP
AnnotationLimitExceeded	Jumlah anotasi objek melebihi batas (1000).	400 Bad Request	Tidak didukung
AnnotationNameTooLong	Nama anotasi Anda terlalu panjang. Nama anotasi dapat mencapai 512 byte panjangnya.	400 Bad Request	Tidak didukung
AnnotationRequestConflict	Operasi yang bertentangan saat ini sedang berlangsung terhadap anotasi ini. Hal ini dapat terjadi ketika anotasi sumber dihapus selama salinan lintas wilayah.	409 Conflict	Tidak didukung
EntityTooSmall	Upload yang Anda usulkan lebih kecil dari ukuran minimum yang diizinkan. Muatan anotasi harus minimal 1 byte.	400 Bad Request	Tidak didukung
InvalidAnnotationName	Nama anotasi Anda berisi karakter yang tidak valid, atau nama dimulai dengan awalan yang dicadangkan (awsatus3, tidak peka huruf besar/kecil). Nama anotasi hanya dapat berisi huruf Unicode, angka, garis bawah (_), titik (.), dan tanda hubung (-).	400 Bad Request	Tidak didukung
InvalidArgument	Arahan anotasi tidak valid. Nilai yang valid adalah COPY dan EXCLUDE. Untuk ListObjectAnnotations, max-annotation-results harus antara 1 dan 1000.	400 Bad Request	Tidak didukung

Kode Kesalahan	Deskripsi	Kode Status HTTP	Awalan Kode Kesalahan SOAP
InvalidPrefix	Awalan anotasi yang Anda berikan tidak valid.	400 Bad Request	Tidak didukung
InvalidRequest	Kesalahan ini mungkin terjadi karena alasan berikut: Objek dienkripsi dengan SSE-C, yang tidak mendukung anotasi. Objek sumber memiliki anotasi yang tidak dapat disalin ke tujuan karena tujuan berada di Wilayah yang tidak didukung. Permintaan konfigurasi tabel anotasi tidak valid (misalnya, menonaktifkan tabel saat menentukan enkripsi, mengaktifkan tabel yang sudah diaktifkan, memperbarui tabel dalam status CREATING, atau mengirimkan permintaan kosong).	400 Bad Request	Tidak didukung
MethodNotAllowed	Metode yang ditentukan tidak diperbolehkan terhadap sumber daya ini. Misalnya, kesalahan ini terjadi ketika permintaan menargetkan penanda hapus.	405 Method Not Allowed	Tidak didukung
NoSuchAnnotation	Anotasi yang ditentukan tidak ada.	404 Not Found	Tidak didukung
NoSuchKey	Kunci yang ditentukan tidak ada.	404 Not Found	Tidak didukung

Kode Kesalahan	Deskripsi	Kode Status HTTP	Awalan Kode Kesalahan SOAP
NoSuchVersion	ID versi yang ditentukan dalam permintaan tidak cocok dengan versi yang ada.	404 Not Found	Tidak didukung
NotImplemented	Anotasi tidak didukung untuk bucket direktori Amazon S3.	501 Not Implemented	Tidak didukung
OperationAborted	Operasi bersyarat yang saling bertentangan saat ini sedang berlangsung terhadap sumber daya ini. Hal ini dapat terjadi selama modifikasi bersamaan atau ketika jumlah anotasi sumber atau ETag berubah selama salinan lintas wilayah. Coba lagi.	409 Conflict	Tidak didukung
PreconditionFailed	Setidaknya salah satu prasyarat yang Anda tentukan tidak berlaku. <code>x-amz-object-if-match</code> ETag tidak cocok dengan objek ETag saat ini.	412 Precondition Failed	Tidak didukung
UnsupportedMediaType	Payload permintaan berisi jenis media yang tidak didukung. Muatan anotasi harus berupa teks yang UTF-8 dikodekan yang valid.	415 Unsupported Media Type	Tidak didukung
UserAnnotationNameMustBeSpecified	Anda harus menentukan nama anotasi pengguna.	400 Bad Request	Tidak didukung

Daftar kode kesalahan Amazon S3 di Outposts

Tabel berikut berisi kesalahan khusus yang mungkin dikembalikan oleh operasi Amazon S3 di Outposts. Untuk informasi umum tentang kesalahan Amazon S3 dan daftar kode kesalahan, lihat [Tanggapan kesalahan](#).

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
BadRequest	Bucket berada dalam keadaan transisi karena upaya penghapusan sebelumnya. Coba lagi nanti.	400 Permintaan Buruk	Tidak didukung
InvalidRequest	Kesalahan ini mungkin terjadi karena alasan berikut: - Konfigurasi Amazon VPC diperlukan. - Akses publik tidak diperbolehkan di S3 pada titik akses Outpost.	400 Permintaan Buruk	Klien
InvalidOutpostState	Permintaan tidak valid untuk status Pos Terdepan saat ini.	409 Konflik	Tidak didukung
InvalidRequest	Titik akses tidak dalam keadaan di mana ia dapat dihapus.	400 Permintaan Buruk	Tidak didukung
NoSuchOutpost	Pos Terdepan yang ditentukan tidak ada.	404 Tidak Ditemukan	Tidak didukung
UnsupportedOperation	Tindakan yang ditentukan tidak didukung.	404 Tidak Ditemukan	Tidak didukung

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
InsufficientCapacity	Kapasitas tidak mencukupi.	507 Penyimpanan Tidak Mencukupi	Tidak didukung

Daftar kode kesalahan Amazon S3 Storage Lens

Tabel berikut berisi kesalahan khusus yang mungkin dikembalikan oleh operasi Amazon S3 Storage Lens. Untuk informasi umum tentang kesalahan Amazon S3 umum dan daftar kode kesalahan, lihat [Tanggapan kesalahan](#).

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
AccessDenied	Wilayah ini tidak didukung sebagai Wilayah rumah untuk S3 Storage Lens.	403 Forbidden	Tidak didukung
AccountNotAuthorized	Akun ini tidak diizinkan untuk digunakan AWS Organizations. Gunakan akun manajemen atau akun administrator yang didelegasikan.	403 Forbidden	Tidak didukung
ActivityMetricsMustEnabled	Metrik aktivitas harus diaktifkan.	400 Permintaan Buruk	Tidak didukung

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
<code>AWSOrganizationsNotInUseException</code>	Akun ini bukan bagian dari organisasi Anda.	403 Forbidden	Tidak didukung
<code>DefaultConfigurationDeleteForbidden</code>	Konfigurasi Default tidak dapat dihapus.	403 Forbidden	Tidak didukung
<code>DuplicateStorageLensGroupARN</code>	Ada dua atau lebih entri dari grup Lensa Penyimpanan yang sama ARN dalam konfigurasi ini.	400 Bad Request	Tidak didukung
<code>EmptyExcludeContainer</code>	Kesalahan ini terjadi karena alasan berikut: • Wadah pengecualian tidak boleh kosong. • Wadah pengecualian tidak dapat memiliki ember nol. • Wadah pengecualian tidak dapat memiliki nol Wilayah.	400 Permintaan Buruk	Tidak didukung
<code>EmptyExcludeElement</code>	Anda harus menentukan grup Lensa Penyimpanan dengan elemen Kecualikan Anda.	400 Bad Request	Tidak didukung

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
EmptyIncludeContainer	Kesalahan ini terjadi karena alasan berikut: • Wadah termasuk tidak boleh kosong. • Wadah termasuk tidak dapat memiliki ember nol. • Wadah include tidak dapat memiliki nol Wilayah.	400 Permintaan Buruk	Tidak didukung
InvalidAWSOrgArn	Ada AWS Organizations ARN yang cacat dalam konfigurasi.	400 Permintaan Buruk	Tidak didukung
EmptyIncludeElement	Anda harus menentukan grup Lensa Penyimpanan dengan elemen Sertakan Anda.	400 Bad Request	Tidak didukung
InvalidBucketFilter	Organization-level konfigurasi tidak mendukung filter bucket.	400 Permintaan Buruk	Tidak didukung
InvalidConfigId	ID konfigurasi tidak valid.	400 Permintaan Buruk	Tidak didukung
InvalidDestination	Bucket S3 ARN cacat.	400 Permintaan Buruk	Tidak didukung

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
InvalidEncryptionMethod	Hanya satu metode enkripsi yang dapat ditentukan.	400 Permintaan Buruk	Tidak didukung
InvalidFilterForDefaultConfiguration	Konfigurasi default tidak boleh menyertakan filter apa pun.	400 Permintaan Buruk	Tidak didukung
InvalidIncludeExcludeContainers	Anda dapat menentukan wadah Sertakan atau wadah Kecualikan dalam konfigurasi. Anda tidak dapat menentukan keduanya dalam konfigurasi.	400 Permintaan Buruk	Tidak didukung
InvalidIncludeExcludeElements	Hanya satu elemen Sertakan atau Kecualikan yang diizinkan. Setidaknya satu elemen Sertakan atau Kecualikan harus ada.	400 Bad Request	Tidak didukung
InvalidKMSEncryptionKeyId	ID kunci KMS ARN tidak valid.	400 Permintaan Buruk	Tidak didukung
InvalidMaximumPrefixDepth	MaxDepth harus berada dalam kisaran [1,10].	400 Permintaan Buruk	Tidak didukung
InvalidMinimumStorageBytesPercentage	MinStorageBytesPercentage harus berada dalam kisaran [1.00,100.00].	400 Permintaan Buruk	Tidak didukung

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
InvalidOrganizationArn	AR AWS Organizations N dalam konfigurasi tidak valid.	400 Permintaan Buruk	Tidak didukung
InvalidOrganizationForDefaultConfiguration	Konfigurasi default tidak mendukung metrik tingkat organisasi.	400 Permintaan Buruk	Tidak didukung
InvalidRegionForDefaultConfiguration	Wilayah yang ditentukan tidak didukung untuk konfigurasi default.	400 Permintaan Buruk	Tidak didukung
InvalidRegionName	Nama Wilayah tidak valid.	400 Permintaan Buruk	Tidak didukung
InvalidStorageLensArn	S3 Storage Lens ARN tidak diperlukan dalam input.	400 Permintaan Buruk	Tidak didukung
InvalidStorageLensGroupArn	Grup Lensa Penyimpanan ARN ini tidak valid atau hanya grup Lensa Penyimpanan di akun Anda yang diizinkan. Selain itu, Anda harus mengikuti struktur ARN grup Lensa Peny arn::s3:::storage-lens-group/impanan: dan mematuhi batas 64 karakter. Nama grup Lensa Penyimpanan juga hanya dapat berisi karakter berikut: a-z,, 0-9 A-Z, tanda hubung (-), dan garis bawah (_).	400 Bad Request	Tidak didukung

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
MissingAccountLevelActivityMetrics	Metrik aktivitas harus diaktifkan di tingkat akun saat metrik aktivitas diaktifkan di tingkat bucket.	400 Permintaan Buruk	Tidak didukung
MissingBucketLevelActivityMetrics	Metrik aktivitas harus diaktifkan di tingkat bucket saat metrik aktivitas diaktifkan di tingkat akun.	400 Permintaan Buruk	Tidak didukung
MissingEncryptionMethod	Metode enkripsi tidak boleh kosong. Tentukan salah satu SSE-KMS atau SSE-S3.	400 Permintaan Buruk	Tidak didukung
MissingPrefixLevelStorageMetrics	Metrik penyimpanan pada tingkat awalan wajib ketika tingkat awalan diaktifkan.	400 Permintaan Buruk	Tidak didukung
OrganizationAccessDenied	Akun ini tidak diizinkan untuk menambahkan AWS Organizations.	403 Dilarang	Tidak didukung
OrgConfigurationNotSupported	Wilayah yang ditentukan tidak mendukung AWS Organizations dalam konfigurasi.	403 Dilarang	Tidak didukung
ServiceNotEnabledForOrg	Peran terkait layanan S3 Storage Lens tidak diaktifkan untuk organisasi.	403 Dilarang	Tidak didukung
StorageMetricsMustEnabled	Prefix-level metrik penyimpanan harus diaktifkan.	400 Permintaan Buruk	Tidak didukung

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
TooManyBuckets	Wadah ember tidak boleh memiliki lebih dari 50 ember.	400 Permintaan Buruk	Tidak didukung
TooManyRegions	Wadah Wilayah tidak boleh memiliki lebih dari 50 Wilayah.	400 Permintaan Buruk	Tidak didukung
TooManyStorageLens Groups	Anda tidak dapat melampirkan lebih dari 50 grup Lensa Penyimpanan ke dasbor Lensa Penyimpanan Anda.	400 Bad Request	Tidak didukung

Tabel berikut berisi kesalahan khusus yang mungkin dikembalikan oleh operasi grup S3 Storage Lens. Untuk informasi umum tentang kesalahan Amazon S3 umum dan daftar kode kesalahan, lihat [Tanggapan kesalahan](#).

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
AccessDenied	Anda tidak memiliki izin untuk melakukan tindakan grup Lensa Penyimpanan. Wilayah ini tidak didukung sebagai Wilayah rumah untuk grup Lensa Penyimpanan S3.	403 Forbidden	Tidak didukung
ConfigurationAlreadyExists	Konfigurasi yang ditentukan sudah ada.	409 Conflict	Tidak didukung

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
DuplicateElement	Tag harus unik. Operator And logis menyertakan kunci tag duplikat. Operator Or logis menyertakan tag duplikat. Operator logis menyertakan prefiks atau akhiran duplikat.	400 Bad Request	Tidak didukung
InvalidAge	DaysLessThan dan DaysGreaterThan harus angka positif.	400 Bad Request	Tidak didukung
InvalidFilter	Filter harus menyertakan salah satu elemen berikut: And, Or, MatchAnyTag, MatchAnyPrefix, MatchAnySuffix, MatchObjectAge, MatchObjectSize.	400 Bad Request	Tidak didukung
InvalidLogicalOperator	Setidaknya dua sub elemen harus ada dalam operator logis And atau Or.	400 Bad Request	Tidak didukung
InvalidMatchAnyPrefix	MatchAnyPrefix Parameter tidak boleh kosong.	400 Bad Request	Tidak didukung
InvalidMatchAnySuffix	MatchAnySuffix Parameter tidak boleh kosong.	400 Bad Request	Tidak didukung
InvalidMatchAnyTag	MatchAnyTag Parameter tidak boleh kosong.	400 Bad Request	Tidak didukung
InvalidMatchObjectAge	MatchObjectAge Parameter tidak boleh kosong.	400 Bad Request	Tidak didukung

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
InvalidMatchObjectSize	MatchObjectSize Parameter tidak boleh kosong.	400 Bad Request	Tidak didukung
InvalidName	Parameter Nama grup Lensa Penyimpanan harus antara 1 dan 64 karakter. NameParameter grup Lensa Penyimpanan harus menggunakan <code>^[a-zA-Z0-9\-_]+\$</code> pola.	400 Bad Request	Tidak didukung
InvalidNumericCombination	Usia objek atau kombinasi ukuran objek ini tidak valid.	400 Bad Request	Tidak didukung
InvalidPrefix	Panjang maksimum awalan adalah 1.024 karakter. String awalan tidak boleh kosong.	400 Bad Request	Tidak didukung
InvalidSize	BytesLessThan dan BytesGreaterThan harus angka positif. Ukuran objek maksimum tidak boleh melebihi 50 TB. Ukuran objek minimum tidak boleh lebih besar dari atau sama dengan 50 TB.	400 Bad Request	Tidak didukung
InvalidSuffix	Panjang maksimum akhiran adalah 1.024 karakter. String akhiran tidak boleh kosong.	400 Bad Request	Tidak didukung

Kode kesalahan	Deskripsi	Kode status HTTP	Awalan kode kesalahan SOAP
InvalidTag	Kunci tag objek tidak boleh melebihi 128 karakter. String kunci tag objek tidak boleh nol atau kosong. Panjang maksimum nilai tanda adalah 256 karakter. Kunci tag objek berisi karakter yang tidak valid. Kunci tag objek harus berisi hanya a-z, A-Z, 0-9, spasi, dan karakter berikut: <code>^(_.: / = + \ - @] *) \$</code>	400 Bad Request	Tidak didukung
MismatchedName	Nama yang ditentukan dalam permintaan tidak cocok dengan nama grup Lensa Penyimpanan.	400 Bad Request	Tidak didukung
TooManyConfigurations	Anda telah mencoba membuat lebih banyak konfigurasi grup Lensa Penyimpanan daripada 50 yang diizinkan.	400 Bad Request	Tidak didukung
TooManyElements	M Element melebihi jumlah maksimum elemen yang diizinkan dalam operator logis. Hanya 10 awalan, sufiks, atau tag yang diperbolehkan.	400 Bad Request	Tidak didukung

Daftar kode kesalahan Lambda Objek Amazon S3

Tabel berikut berisi kesalahan khusus yang mungkin dikembalikan S3 Object Lambda. Untuk informasi tentang kesalahan Amazon S3 umum dan daftar kode kesalahan, lihat [Tanggapan kesalahan](#).

Tanggapan kesalahan yang diterima dari titik akses pendukung selama non- GetObject permintaan dikirim ke penelepon tanpa diubah.

Kode kesalahan	Deskripsi	Kode status HTTP	
LambdaInvalidResponse	Kembali ke penelepon asli saat WriteGetObjectResponse merespons ValidationError dengan AWS Lambda. Lihat ValidationError pesan untuk lebih jelasnya. Tidak semua kasus ValidationError menghasilkan LambdaInvalidResponse kesalahan.	400 Bad Request	
LambdaInvocationFailed	Pemanggilan fungsi lambda gagal. Pemanggil mungkin menerima kesalahan berikut ketika S3 Object Lambda tidak dapat berhasil memanggil fungsi Lambda yang dikonfigurasi. Pesan kesalahan mungkin berisi detail tentang kesalahan akhirnya yang dikembalikan oleh AWS Lambda layanan saat memanggil fungsi (misalnya, kode status, kode kesalahan, pesan kesalahan, dan ID permintaan).	400 Bad Request	
LambdaNotFound	AWS Lambda Fungsi tidak ditemukan. Fungsi, versi, atau alias Lambda yang dikonfigurasi tidak ditemukan saat mencoba memanggilnya.	404 Not Found	

Kode kesalahan	Deskripsi	Kode status HTTP	
	Pastikan konfigurasi S3 Object Lambda Access Point menunjuk ke fungsi Lambda ARN yang benar. Pesan kesalahan mungkin berisi detail tentang kesalahan akhirnya yang dikembalikan oleh AWS Lambda layanan saat memanggil fungsi (misalnya, kode status, kode kesalahan, pesan kesalahan, dan ID permintaan).		
LambdaPermissionError	Pemanggil tidak berwenang untuk memanggil fungsi Lambda. Pemanggil harus memiliki izin untuk memanggil fungsi Lambda. Periksa kebijakan yang dilampirkan ke pemanggil dan pastikan bahwa kebijakan tersebut telah diizinkan untuk digunakan <code>lambda:Invoke</code> untuk fungsi yang dikonfigurasi. Pesan kesalahan mungkin berisi detail tentang kesalahan akhirnya yang dikembalikan oleh AWS Lambda layanan saat memanggil fungsi (misalnya, kode status, kode kesalahan, pesan kesalahan, dan ID permintaan).	403 Forbidden	

Kode kesalahan	Deskripsi	Kode status HTTP	
LambdaResponseNotReceived	Fungsi Lambda keluar tanpa berhasil memanggil <code>WriteGetObjectResponse</code>. <code>GetObject</code> data respons disediakan oleh fungsi Lambda dengan memanggil operasi <code>WriteGetObjectResponse</code> API. CloudWatch Log Amazon untuk fungsi tersebut mungkin memberikan wawasan lebih lanjut tentang mengapa fungsi tidak berhasil memanggil operasi API ini meskipun keluar secara normal.	500 Internal Service Error	
LambdaRuntimeError	Fungsi Lambda gagal selama eksekusi. Kesalahan eksplisit diterima dari fungsi Lambda. Untuk detail tentang kegagalan, periksa AWS CloudFormation log.	500 Internal Service Error	
LambdaTimeout	Fungsi Lambda tidak merespons dalam waktu yang diizinkan. Fungsi Lambda gagal menyelesaikan panggilannya <code>WriteGetObjectResponse</code> dalam waktu 60 detik.	500 Internal Service Error	

Kode kesalahan	Deskripsi	Kode status HTTP	
SlowDown	Kurangi tingkat permintaan Anda untuk operasi yang melibatkan AWS Lambda. Pemanggilan fungsi dibatasi oleh AWS Lambda, mungkin karena telah mencapai batasan konkurensi yang dikonfigurasi. Untuk informasi selengkapnya, lihat Mengelola konkurensi untuk fungsi Lambda di Panduan Peng AWS Lambda embang. Pesan kesalahan mungkin berisi detail tentang kesalahan akhirnya yang dikembalikan oleh AWS Lambda layanan saat memanggil fungsi (misalnya, kode status, kode kesalahan, pesan kesalahan, dan ID permintaan).	503 Slow Down	
ValidationError	Kesalahan validasi mungkin dikembalikan dari operasi WriteGetObjectResponse API dan dapat terjadi karena berbagai alasan. Lihat pesan kesalahan untuk detail selengkapnya.	400 Bad Request	

Daftar kode kesalahan asinkron Amazon S3

Tabel berikut berisi kesalahan khusus yang mungkin dikembalikan oleh permintaan asinkron. Untuk informasi umum tentang kesalahan Amazon S3 dan daftar kode kesalahan, lihat [Tanggapan kesalahan](#).

Kesalahan ini dikembalikan ketika Anda menanyakan tentang status permintaan asinkron, seperti dengan menggunakan `DescribeMultiRegionAccessPointOperation`. Karena permintaan ini tidak sinkron, semua kesalahan ini memiliki kode status `200 OK`.

Kode kesalahan	Deskripsi	Kode status HTTP	
<code>AccessDenied</code>	Akses ditolak.	<code>200 OK</code>	
<code>InternalErrors</code>	Terjadi kesalahan server internal.	<code>200 OK</code>	
<code>MalformedPolicy</code>	Sintaks kebijakan yang ditentukan tidak valid.	<code>200 OK</code>	
<code>MultiRegionAccessPointAlreadyOwnedByYou</code>	Anda sudah memiliki Multi-Region Access Point dengan nama yang sama.	<code>200 OK</code>	
<code>MultiRegionAccessPointModifiedByAnotherRequest</code>	Tindakan gagal karena permintaan lain memodifikasi sumber daya yang ditentukan. Coba kirim ulang permintaan Anda setelah permintaan sebelumnya selesai.	<code>200 OK</code>	
<code>MultiRegionAccessPointNotReady</code>	Titik Multi-Region Akses yang ditentukan belum siap untuk diperbarui.	<code>200 OK</code>	
<code>MultiRegionAccessPointSameBucketRegion</code>	Bucket yang digunakan untuk membuat Multi-Region Access Point tidak boleh berada di Wilayah yang sama.	<code>200 OK</code>	
<code>MultiRegionAccessPointUnsupportedRegion</code>	Salah satu bucket yang disediakan untuk membuat Multi-Region Access Point berada di Wilayah yang tidak didukung.	<code>200 OK</code>	
<code>NoSuchBucket</code>	Bucket yang ditentukan tidak ada.	<code>200 OK</code>	

Kode kesalahan	Deskripsi	Kode status HTTP	
NoSuchMultiRegionAccessPoint	Titik Multi-Region Akses yang ditentukan tidak ada.	200 OK	

Daftar kode kesalahan Amazon S3 Access Grants

Tabel berikut berisi kesalahan khusus yang mungkin dikembalikan oleh permintaan S3 Access Grants. Untuk informasi umum tentang kesalahan Amazon S3 dan daftar kode kesalahan, lihat [Tanggapan kesalahan](#).

Kode Kesalahan	Deskripsi	Kode Status HTTP	
AccessGrantAlreadyExists	Hibah akses yang ditentukan sudah ada	409	
AccessGrantsInstanceAlreadyExists	Instance Access Grants sudah ada	409	
AccessGrantsInstanceNotEmptyError	Harap bersihkan lokasi sebelum menghapus instance akses hibah	400	
AccessGrantsInstanceDoesNotExistError	Instans Access Grants tidak ada	404	
AccessGrantsInstanceResourcePolicyDoesNotExist	Kebijakan Sumber Daya Instance Access Grants tidak ada	404	
AccessGrantsLocationAlreadyExistsError	Lokasi pemberian akses yang ditentukan sudah ada	409	
AccessGrantsLocationNotEmptyError	Harap bersihkan hibah akses sebelum menghapus lokasi hibah akses	400	

Kode Kesalahan	Deskripsi	Kode Status HTTP	
AccessGrantsLocationsQuotaExceededError	Kuota lokasi pemberian akses telah terlampaui. Kuota Lokasi Hibah Akses:<value>. Silakan hubungi S3 jika diperlukan peningkatan.	409	
AccessGrantsQuotaExceededError	Kuota hibah akses telah terlampaui. Kuota Hibah Akses:<value>. Silakan hubungi S3 jika diperlukan peningkatan.	409	
InvalidTag	Ada kunci tag duplikat dalam permintaan Anda. Hapus kunci tag duplikat dan coba lagi.	400	
InvalidAccessGrant	Hibah Akses yang ditentukan tidak valid	400	
InvalidAccessGrantLocation	Lokasi Access Grants yang ditentukan tidak valid	400	
InvalidIamRole	Peran IAM yang ditentukan tidak valid	400	
InvalidIdentityCenterInstance	Instance pusat identitas yang ditentukan tidak valid	400	
InvalidResourcePolicy	Kebijakan Sumber Daya yang ditentukan tidak valid	400	
InvalidResourcePolicy	Kebijakan Sumber Daya yang ditentukan tidak valid	400	

Kode Kesalahan	Deskripsi	Kode Status HTTP	
InvalidTag	Permintaan ini berisi kunci tag atau nilai yang tidak valid. Karakter yang valid meliputi: [a-z A-Z +-=._:/]. Kunci tag dapat berisi hingga 128 karakter. Nilai tag dapat berisi hingga 256 karakter.	400	
NoSuchAccessGrantError	Hibah akses yang ditentukan tidak ada	404	
NoSuchAccessGrantsLocationError	Lokasi pemberian akses yang ditentukan tidak ada	404	
AccessDenied	Anda tidak memiliki <i><requested permission></i> izin untuk Awalan S3 yang diminta: <i><requested target></i>	403 Forbidden	
StsNotAuthorizedError	Terjadi kesalahan (StsNotAuthorizedError) saat memanggil GetDataAccess operasi: Pengguna: access-grants.s3.amazonaws.com tidak berwenang untuk melakukan: sts:AssumeRole pada sumber daya: <i><IAM Role ARN></i>	403	
StsPackedPolicyTooLargeError	Terjadi kesalahan (StsPackedPolicyTooLargeError) saat memanggil GetDataAccess operasi: Token serial terlalu besar untuk sesi	400	
StsValidationError	Pesan kesalahan bervariasi tergantung pada kesalahan validasi.	400	

Kode Kesalahan	Deskripsi	Kode Status HTTP	
InvalidTags	Kunci tag tidak dapat dimulai dengan aw AWS alan yang dicadangkan untuk tag sistem.”	400	
TooManyTags	Jumlah tag melebihi batas 50 tag. Hapus beberapa tag dan coba lagi.	400	

Daftar kode FSx-related kesalahan Amazon

Tabel berikut berisi kesalahan khusus yang mungkin dikembalikan oleh permintaan yang dibuat ke Amazon FSx melalui titik akses Amazon S3. Untuk informasi umum tentang kesalahan Amazon S3 dan daftar kode kesalahan, lihat [Tanggapan kesalahan](#).

Kode Kesalahan	Deskripsi	Kode Status HTTP	
AccessDenied	Kesalahan ini mungkin terjadi karena alasan berikut: - Penelepon tidak berwenang untuk melakukan operasi S3 yang ditentukan. - Penelepon tidak berwenang untuk mengakses sumber daya Amazon FSx untuk kunci yang ditentukan.	403 Forbidden	
InsufficientCapacity	Kapasitas penyimpanan maksimum sistem file telah tercapai.	507 Insufficient Capacity	

Kode Kesalahan	Deskripsi	Kode Status HTTP	
InvalidKey	Kunci yang ditentukan tidak valid.	400 Bad Request	
SlowDown	Harap kurangi tarif permintaan Anda.	503 Slow Down	

Daftar kode kesalahan Amazon S3 Tables

Tabel berikut berisi kesalahan khusus yang mungkin dikembalikan oleh operasi Amazon S3 Tables. Untuk informasi umum tentang kesalahan Amazon S3 dan daftar kode kesalahan, lihat [Tanggapan kesalahan](#).

Kode Kesalahan	Deskripsi	Kode Status HTTP
AccessDeniedException	Tindakan tidak dapat dilakukan karena Anda tidak memiliki izin yang diperlukan.	403 Forbidden
BadRequestException	Permintaan tidak valid atau cacat.	400 Bad Request
ConflictException	Permintaan gagal karena ada konflik dengan penulisan sebelumnya. Anda dapat mencoba kembali permintaan tersebut.	409 Conflict
ForbiddenException	Penelepon tidak berwenang untuk membuat permintaan.	403 Forbidden
InternalServerErrorException	Permintaan gagal karena kesalahan server internal.	500 Internal Server Error

Kode Kesalahan	Deskripsi	Kode Status HTTP
NotFoundException	Permintaan ditolak karena sumber daya yang ditentukan tidak dapat ditemukan.	404 Not Found
TooManyRequestsException	Batas jumlah permintaan per detik terlampaui.	429 Too Many Requests

Praktik terbaik kesalahan Amazon S3

Banyak respons kesalahan berisi data terstruktur tambahan yang dimaksudkan untuk dibaca dan dipahami oleh pengembang yang mendiagnosis kesalahan pemrograman. Misalnya, jika Anda mengirim Content-MD5 header dengan permintaan REST PUT yang tidak cocok dengan ringkasan yang dihitung di server, Anda menerima BadDigest kesalahan. Respons kesalahan juga mencakup sebagai elemen detail ringkasan yang dihitung, dan intisari yang Anda berikan untuk diharapkan. Selama pengembangan, Anda dapat menggunakan informasi ini untuk mendiagnosis kesalahan. Dalam produksi, program yang berperilaku baik mungkin menyertakan informasi ini dalam log kesalahannya.

Saat merancang aplikasi untuk digunakan dengan Amazon S3, penting untuk menangani kesalahan Amazon S3 sebagaimana mestinya. Bagian ini menjelaskan masalah yang perlu dipertimbangkan saat merancang aplikasi Anda.

Coba lagi InternalErrors

Kesalahan internal adalah kesalahan yang terjadi dalam lingkungan Amazon S3.

Permintaan yang menerima InternalError tanggapan mungkin belum diproses. Misalnya, jika permintaan PUT kembali InternalError, GET berikutnya mungkin mengambil nilai lama atau nilai yang diperbarui.

Jika Amazon S3 mengembalikan InternalError respons, coba lagi permintaan tersebut.

Setel aplikasi untuk SlowDown kesalahan berulang

Seperti halnya sistem terdistribusi lainnya, S3 memiliki mekanisme perlindungan yang mendeteksi konsumsi sumber daya yang disengaja atau tidak disengaja dan bereaksi sesuai dengan itu.

SlowDown kesalahan dapat terjadi ketika tingkat permintaan tinggi memicu salah satu mekanisme ini. Mengurangi tingkat permintaan Anda akan mengurangi atau menghilangkan kesalahan tipe ini. Secara umum, sebagian besar pengguna tidak akan mengalami kesalahan ini secara teratur; Namun, jika Anda ingin informasi lebih lanjut atau mengalami SlowDown kesalahan tinggi atau tidak terduga, silakan kirim ke forum pengembang [Amazon S3 kami](#) atau daftar Dukungan <https://aws.amazon.com/premiumsupport/>.

Isolasi kesalahan

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

Amazon S3 menyediakan serangkaian kode kesalahan yang digunakan oleh API SOAP dan REST. SOAP API mengembalikan kode kesalahan Amazon S3 standar. API REST dirancang untuk terlihat seperti server HTTP standar dan berinteraksi dengan klien HTTP yang ada (mis. peramban, pustaka klien HTTP, proksi, cache, dan lain-lain). Untuk memastikan klien HTTP menangani kesalahan dengan benar, Amazon S3 memetakan setiap kesalahan Amazon S3 ke kode status HTTP.

Kode status HTTP kurang ekspresif dibandingkan kode kesalahan Amazon S3 dan berisi lebih sedikit informasi tentang kesalahan tersebut. Misalnya, kesalahan NoSuchKey dan NoSuchBucket Amazon S3 keduanya memetakan ke kode status HTTP 404 Not Found.

Meskipun kode status HTTP memuat lebih sedikit informasi tentang kesalahan, klien yang memahami HTTP, tetapi bukan API Amazon S3, biasanya akan menangani kesalahan dengan benar.

Oleh karena itu, saat menangani kesalahan atau melaporkan kesalahan Amazon S3 kepada pengguna akhir, gunakan kode kesalahan Amazon S3 alih-alih kode status HTTP karena kode ini berisi sebagian besar informasi tentang kesalahan tersebut. Selain itu, saat men-debug aplikasi Anda, Anda juga harus melihat elemen <Details> yang dapat dibaca manusia dari respons kesalahan XML.

AWS Glosarium

Untuk AWS terminologi terbaru, lihat [AWS glosarium di Referensi](#).Glosarium AWS

Sumber Daya Amazon S3

Berikut ini adalah tabel yang mencantumkan daftar sumber daya terkait yang akan berguna saat Anda bekerja dengan layanan ini.

Sumber Daya	Deskripsi
Panduan Pengguna Layanan Penyimpanan Sederhana Amazon	Panduan memulai memberikan tutorial cepat layanan berdasarkan kasus penggunaan sederhana.
Panduan Pengguna Layanan Penyimpanan Sederhana Amazon	Panduan pengembang menjelaskan cara menyelesaikan tugas menggunakan operasi Amazon S3.
FAQ Teknis Amazon S3	FAQ mencakup 20 pertanyaan teratas yang diajukan pengembang tentang produk ini.
Catatan Rilis Amazon S3	Catatan Rilis memberikan gambaran tingkat tinggi dari rilis saat ini. Secara khusus, catatan tersebut mencatat fitur, koreksi, dan masalah yang diketahui baru.
Alat untuk Amazon Web Services	Titik awal utama untuk menemukan dokumentasi, contoh kode, catatan rilis, dan informasi lainnya untuk membantu Anda membangun aplikasi inovatif dengan AWS SDK dan alat.
Konsol Manajemen AWS	Konsol memungkinkan Anda untuk melakukan sebagian besar fungsi Amazon S3 tanpa pemrograman.
Forum Diskusi	Forum berbasis komunitas bagi pengembang untuk membahas pertanyaan teknis yang terkait dengan Amazon Web Services.
Pusat Dukungan	Halaman beranda untuk AWS Dukungan Teknis, termasuk akses ke Forum Pengembang, FAQ Teknis, halaman Status Layanan, dan Dukungan Premium.
Dukungan	Halaman web utama untuk informasi tentang Dukungan, saluran dukungan respons cepat satu-satu untuk

Sumber Daya	Deskripsi
	membantu Anda membangun dan menjalankan aplikasi di Layanan AWS Infrastruktur.
Informasi produk Amazon S3	Halaman web utama untuk informasi tentang Amazon S3.
Hubungi Kami	Titik kontak pusat untuk pertanyaan mengenai AWS penagihan, akun, peristiwa, penyalahgunaan, dll.
Ketentuan Penggunaan	Informasi terperinci tentang penggunaan hak cipta dan merek dagang di Amazon.com dan topik lainnya.

Lampiran

Topik

- [Lampiran: Tang SelectObjectContent gapan](#)
- [Lampiran: objek OPTIONS](#)
- [Lampiran: SOAP API](#)
- [Lampiran: Permintaan otentikasi \(AWS tanda tangan versi 2\)](#)
- [Lampiran: API Konfigurasi Siklus Hidup \(Tidak Digunakan\)](#)

Lampiran: Tang SelectObjectContent gapan

Deskripsi

Operasi Amazon S3 Select menyaring konten objek Amazon S3 berdasarkan pernyataan bahasa kueri terstruktur (SQL) sederhana. Mengingat ukuran respons operasi ini tidak diketahui, Amazon S3 Select mengalirkan respons sebagai serangkaian pesan dan menyer Transfer-Encoding takan header dengan **chunked** nilainya dalam respons.

Untuk informasi selengkapnya tentang Amazon S3 Select, lihat [Memilih Konten dari Objek](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Untuk informasi selengkapnya tentang menggunakan SQL dengan Amazon S3 Select, lihat Referensi [SQL untuk Amazon S3 Select dan Amazon Glacier Select](#) di Panduan Pengguna Amazon Simple Storage Service.

Respons

Operasi Select Amazon S3 yang berhasil mengembalikan kode 200 OK status.

Header Respons

Implementasi operasi ini hanya menggunakan header respons yang umum untuk sebagian besar respons. Untuk informasi selengkapnya, lihat [Header Respons Umum](#).

Isi Respons

Karena ukuran respons Amazon S3 Select tidak diketahui, Amazon S3 mengalirkan respons sebagai serangkaian pesan dan menyer Transfer-Encoding takan header dengan **chunked** nilainya dalam respons. Contoh berikut menunjukkan format respons di tingkat atas:

```
<Message 1>
<Message 2>
<Message 3>
.....
<Message n>
```

Setiap pesan terdiri dari dua bagian: pendahuluan dan data. Bagian pendahuluan terdiri dari 1) panjang byte total pesan, dan 2) panjang byte gabungan dari semua header. Bagian data terdiri dari 1) header, dan 2) muatan.

Setiap bagian diakhiri dengan checksum integer big-endian 4-byte (CRC). Amazon S3 Select menggunakan CRC32 (sering disebut sebagai GZIP CRC32) untuk menghitung kedua CRC. Untuk informasi selengkapnya tentang CRC32, lihat spesifikasi format file [GZIP versi 4.3](#).

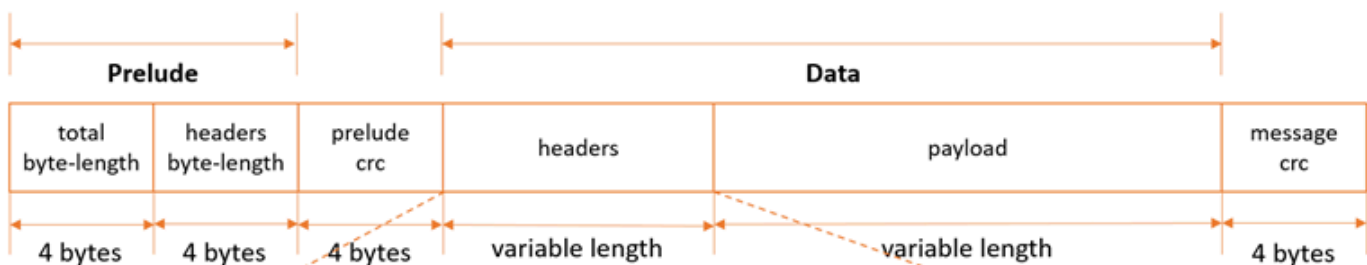
Total overhead pesan termasuk pendahuluan dan kedua checksum adalah 16 byte.

Note

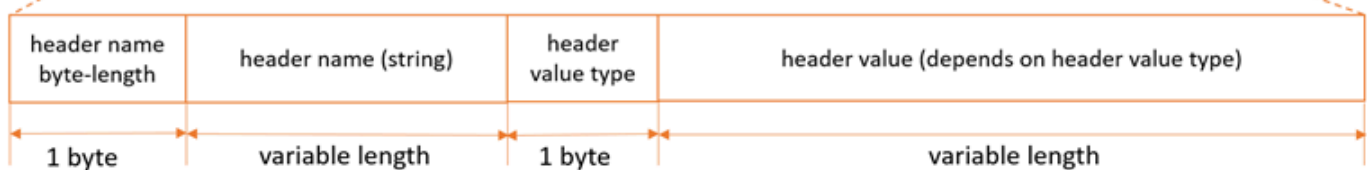
Semua integer nilai dalam pesan berada dalam urutan byte jaringan, atau urutan big-endian.

Diagram berikut menunjukkan komponen yang membentuk pesan dan header. Perhatikan bahwa ada beberapa header per pesan.

Message:



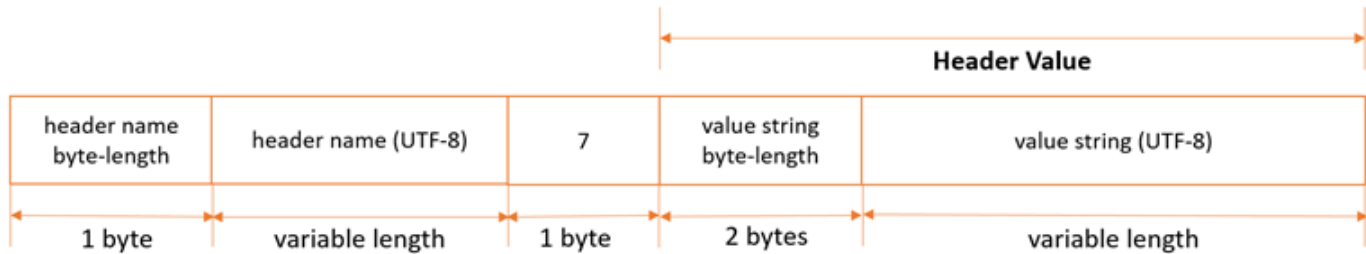
Headers (multiple headers per message):



Note

Untuk Amazon S3 Select, tipe nilai header selalu 7 (type=string). Untuk tipe ini, nilai header terdiri dari dua komponen, panjang bilangan bulat big-endian 2-byte, dan UTF-8 string yang memiliki panjang byte tersebut. Diagram berikut menunjukkan komponen yang membentuk header Amazon S3 Select.

Amazon S3 Select Headers (type=String):



Perhitungan panjang byte muatan (dua perhitungan ini setara):

- $\text{payload\_length} = \text{total\_length} - \text{header\_length} - \text{SizeOf}(\text{total\_length}) - \text{SizeOf}(\text{header\_length}) - \text{SizeOf}(\text{prelude\_crc}) - \text{SizeOf}(\text{pesan\_crc})$
- $\text{payload\_length} = \text{total\_length} - \text{header\_length} - 16$

Setiap pesan berisi komponen-komponen berikut:

- Pendahuluan: Selalu tetap ukuran 8 byte (dua bidang masing-masing 4 byte):
 - Empat byte pertama: Total panjang byte: panjang byte Big-endian integer dari seluruh pesan (termasuk bidang panjang total 4-byte itu sendiri).
 - Empat byte kedua: Header byte-length: Big-endian integer byte-length dari bagian header pesan (tidak termasuk bidang panjang header itu sendiri).
- Prelude CRC: 4-byte big-endian integer checksum (CRC) untuk bagian awal pesan (tidak termasuk CRC itu sendiri). Pendahuluan memiliki CRC terpisah dari pesan CRC (lihat di bawah), untuk memastikan bahwa informasi panjang byte yang rusak dapat dideteksi segera, tanpa menyebabkan perilaku buffering patologis.
- Header: Satu set metadata yang membuat anotasi pesan, seperti jenis pesan, format payload, dan sebagainya. Pesan dapat memiliki beberapa header, sehingga bagian pesan ini dapat memiliki panjang byte yang berbeda tergantung pada jenis pesan. Header adalah pasangan kunci-nilai, di mana kunci dan nilai adalah string. UTF-8 Header dapat muncul dalam urutan apa pun dalam bagian header pesan, dan setiap jenis header yang diberikan hanya dapat muncul sekali.

Untuk Amazon S3 Select, berikut adalah daftar nama header dan kumpulan nilai yang valid tergantung pada jenis pesan.

- MessageType Header:

- HeaderName => “:jenis pesan”
- Val HeaderValues id => “kesalahan”, “acara”
- EventType Header:
 - HeaderName => “:tipe-acara”
 - Val HeaderValues id => “Catatan”, “Konten”, “Kemajuan”, “Statistik”, “Akhir”
- ErrorCode Header:
 - HeaderName => “:kode kesalahan”
 - Val HeaderValues id => Kode Kesalahan dari tabel di [Daftar Kode Kesalahan Konten Objek SELECT](#) bagian.
- ErrorMessage Header:
 - HeaderName => “:pesan kesalahan”
 - Val HeaderValues id => Pesan kesalahan yang dikembalikan oleh layanan, untuk membantu mendiagnosis kesalahan tingkat permintaan.
- Payload: Bisa apa saja.
- Pesan CRC: 4-byte big-endian integer checksum (CRC) dari awal pesan hingga awal checksum (yaitu, semua yang ada di pesan tidak termasuk pesan CRC itu sendiri).

Setiap header berisi komponen-komponen berikut. Mungkin ada beberapa header per pesan.

- Nama Header Byte-Length: Byte-length dari nama header.
- Nama Header: Nama header, menunjukkan jenis header. Nilai yang valid: “:message-type” “:event-type” “:error-code” “:error-message”
- Jenis Nilai Header: Enum menunjukkan jenis nilai header. Untuk Amazon S3 Select, ini selalu 7.
- Nilai String Byte-Length: (Untuk Amazon S3 Select) Byte-length dari string nilai header.
- String Nilai Header: (Untuk Amazon S3 Pilih) Nilai string header. Nilai yang valid untuk bidang ini bervariasi berdasarkan jenis header. Lihat bagian di bawah ini untuk nilai yang valid untuk setiap jenis header dan jenis pesan.

Untuk Amazon S3 Select, tanggapan dapat berupa pesan dari jenis berikut:

- Pesan rekaman: Dapat berisi satu catatan, catatan paruh, atau beberapa catatan. Bergantung pada ukuran hasilnya, respons dapat berisi satu atau lebih pesan ini.

- Pesan lanjutan: Amazon S3 secara berkala mengirimkan pesan ini untuk menjaga koneksi TCP tetap terbuka. Pesan-pesan ini muncul dalam tanggapan secara acak. Klien harus mendeteksi jenis pesan dan memproses yang sesuai.
- Pesan kemajuan: Amazon S3 secara berkala mengirimkan pesan ini, jika diminta. Ini berisi informasi tentang kemajuan kueri yang telah dimulai tetapi belum selesai.
- Pesan statistik: Amazon S3 mengirimkan pesan ini di akhir permintaan. Ini berisi statistik tentang kueri.
- Pesan akhir: Menunjukkan bahwa permintaan selesai, dan tidak ada lagi pesan yang akan dikirim. Anda tidak boleh berasumsi bahwa permintaan selesai sampai klien menerima End pesan.
- RequestLevelError pesan: Amazon S3 mengirimkan pesan ini jika permintaan gagal karena alasan apa pun. Ini berisi kode kesalahan dan pesan kesalahan untuk kegagalan. Jika Amazon S3 mengirim RequestLevelError pesan, itu tidak mengirim End pesan.

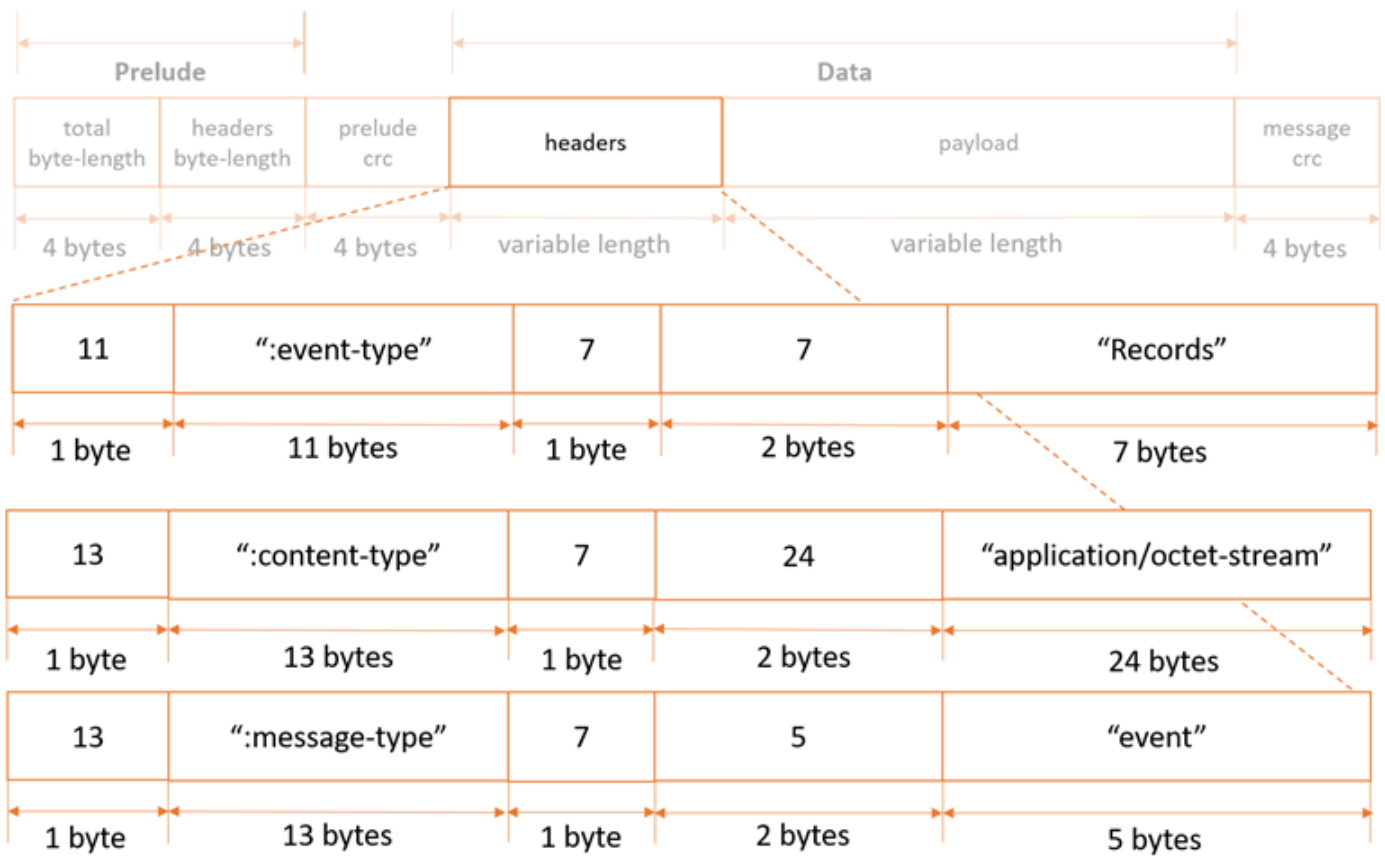
Bagian berikut menjelaskan struktur setiap jenis pesan secara lebih rinci.

Untuk contoh kode dan pengujian unit yang menggunakan protokol ini, lihat [AWS C Event Stream](#) di GitHub situs web.

Rekaman Pesan

Spesifikasi header

Pesan rekaman berisi tiga header, sebagai berikut:



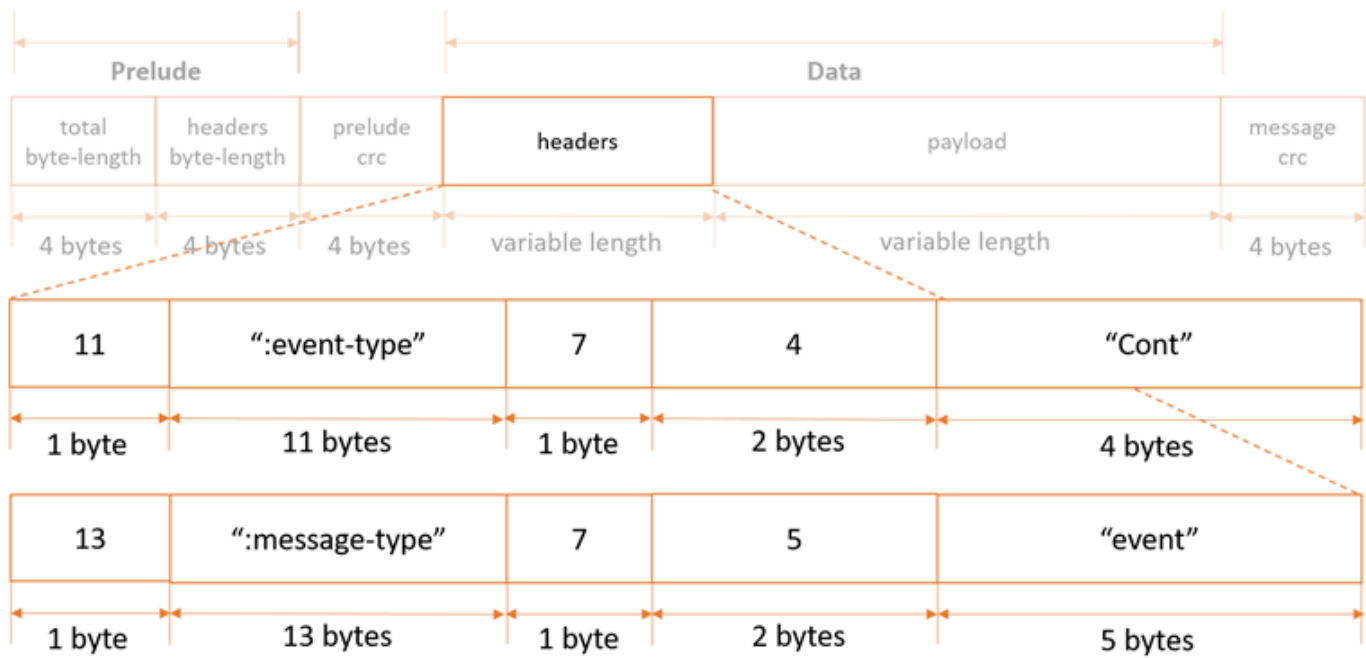
Spesifikasi muatan

Payload pesan rekaman dapat berisi satu catatan, catatan parsional, atau beberapa catatan.

Pesan Kelanjutan

Spesifikasi header

Pesan lanjutan berisi dua header, sebagai berikut:



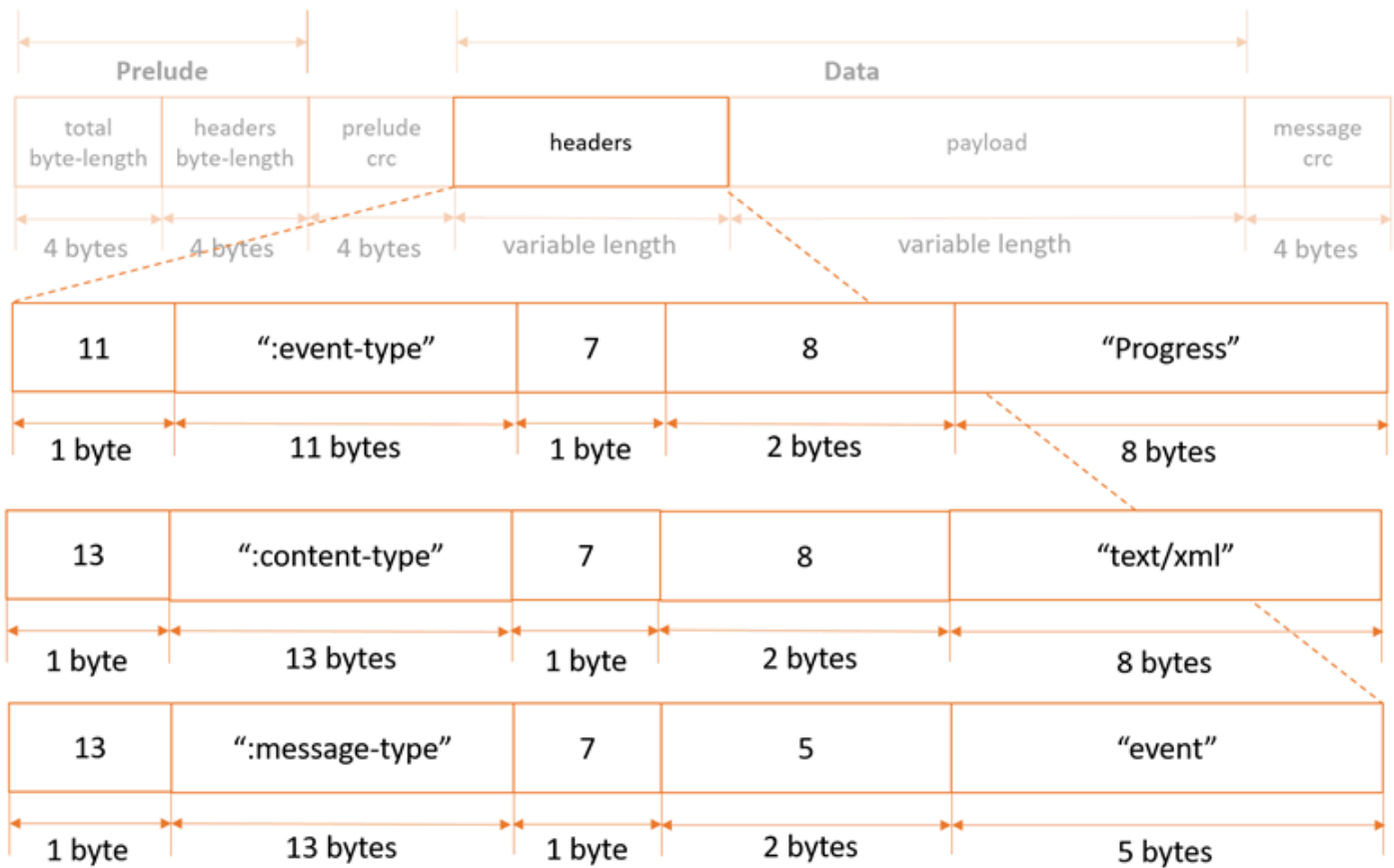
Spesifikasi muatan

Pesan lanjutan tidak memiliki muatan.

Pesan Kemajuan

Spesifikasi header

Pesan kemajuan berisi tiga header, sebagai berikut:



Spesifikasi muatan

Payload pesan kemajuan adalah dokumen XML yang berisi informasi tentang kemajuan permintaan.

- BytesScanned=> Jumlah byte yang telah diproses sebelum dikompresi (jika file dikompresi).
- BytesProcessed=> Jumlah byte yang telah diproses setelah tidak dikompresi (jika file dikompresi).
- BytesReturned=> Jumlah byte data muatan rekaman saat ini yang dikembalikan oleh Amazon S3.

Untuk file yang tidak terkompresi, BytesScanned dan BytesProcessed sama.

Contoh:

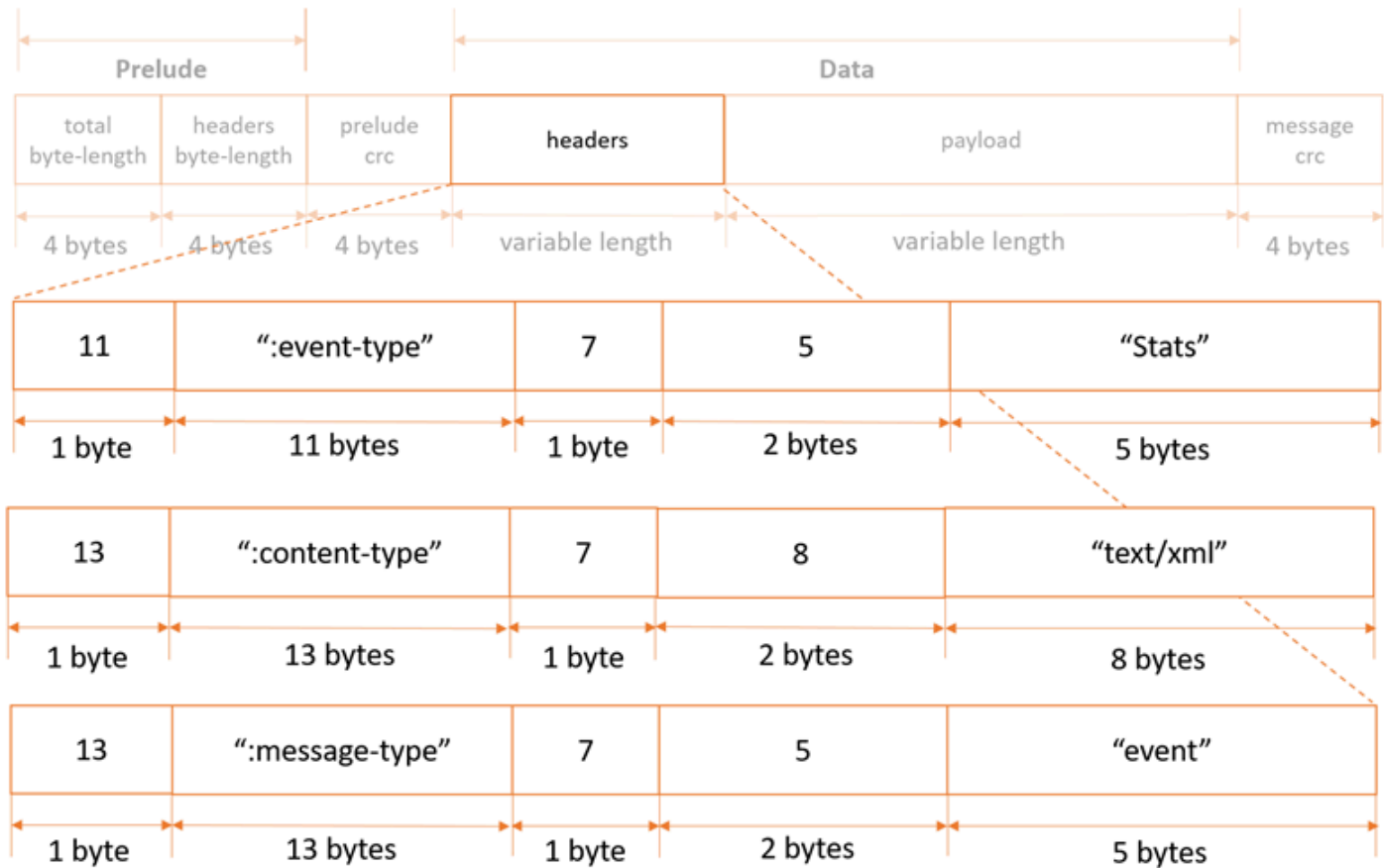
```
<?xml version="1.0" encoding="UTF-8"?>
<Progress>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
```


</Progress>

Pesan Statistik

Spesifikasi header

Pesan statistik berisi tiga header, sebagai berikut:



Spesifikasi muatan

Payload pesan statistik adalah dokumen XML yang berisi informasi tentang statistik permintaan saat pemrosesan selesai.

- BytesScanned=> Jumlah byte yang telah diproses sebelum dikompresi (jika file dikompresi).
- BytesProcessed=> Jumlah byte yang telah diproses setelah tidak dikompresi (jika file dikompresi).
- BytesReturned=> Jumlah total byte data muatan rekaman yang dikembalikan oleh Amazon S3.

Untuk file yang tidak terkompresi, BytesScanned dan BytesProcessed sama.

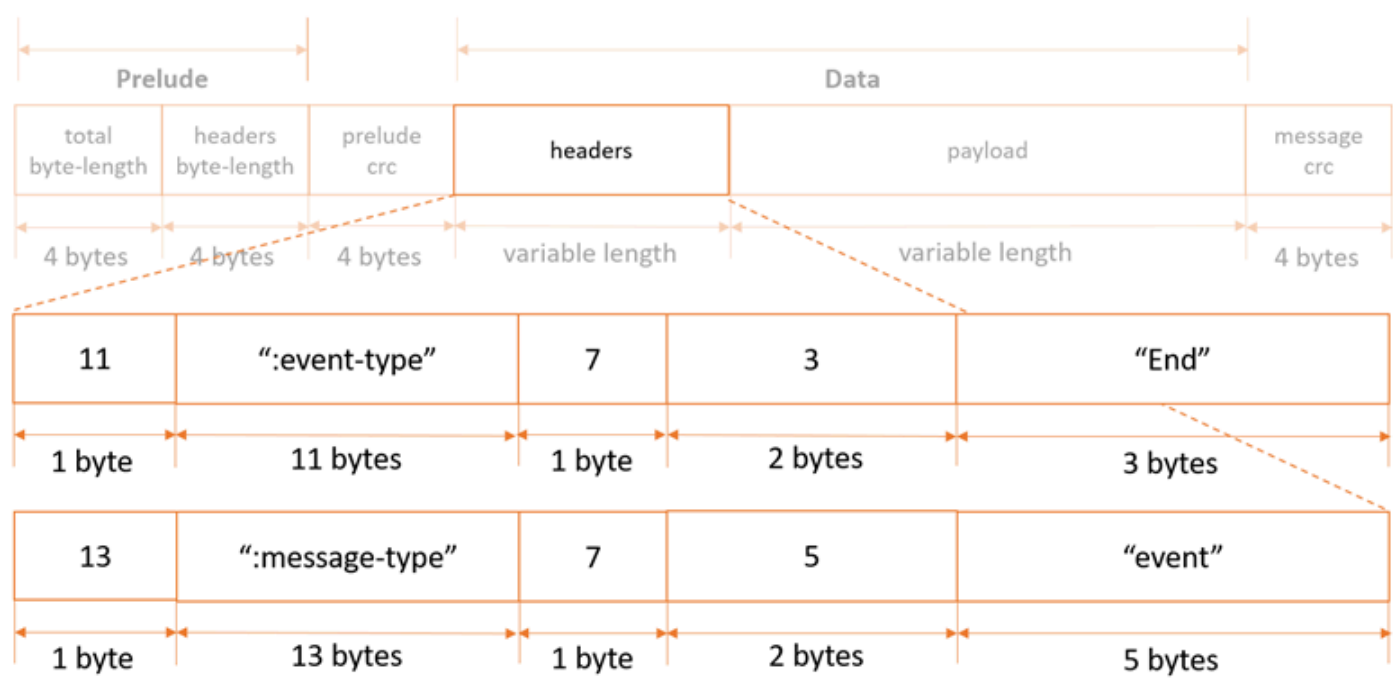
Contoh:

```
<?xml version="1.0" encoding="UTF-8"?>
<Stats>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
</Stats>
```

Pesan Akhir

Spesifikasi header

Pesan akhir berisi dua header, sebagai berikut:



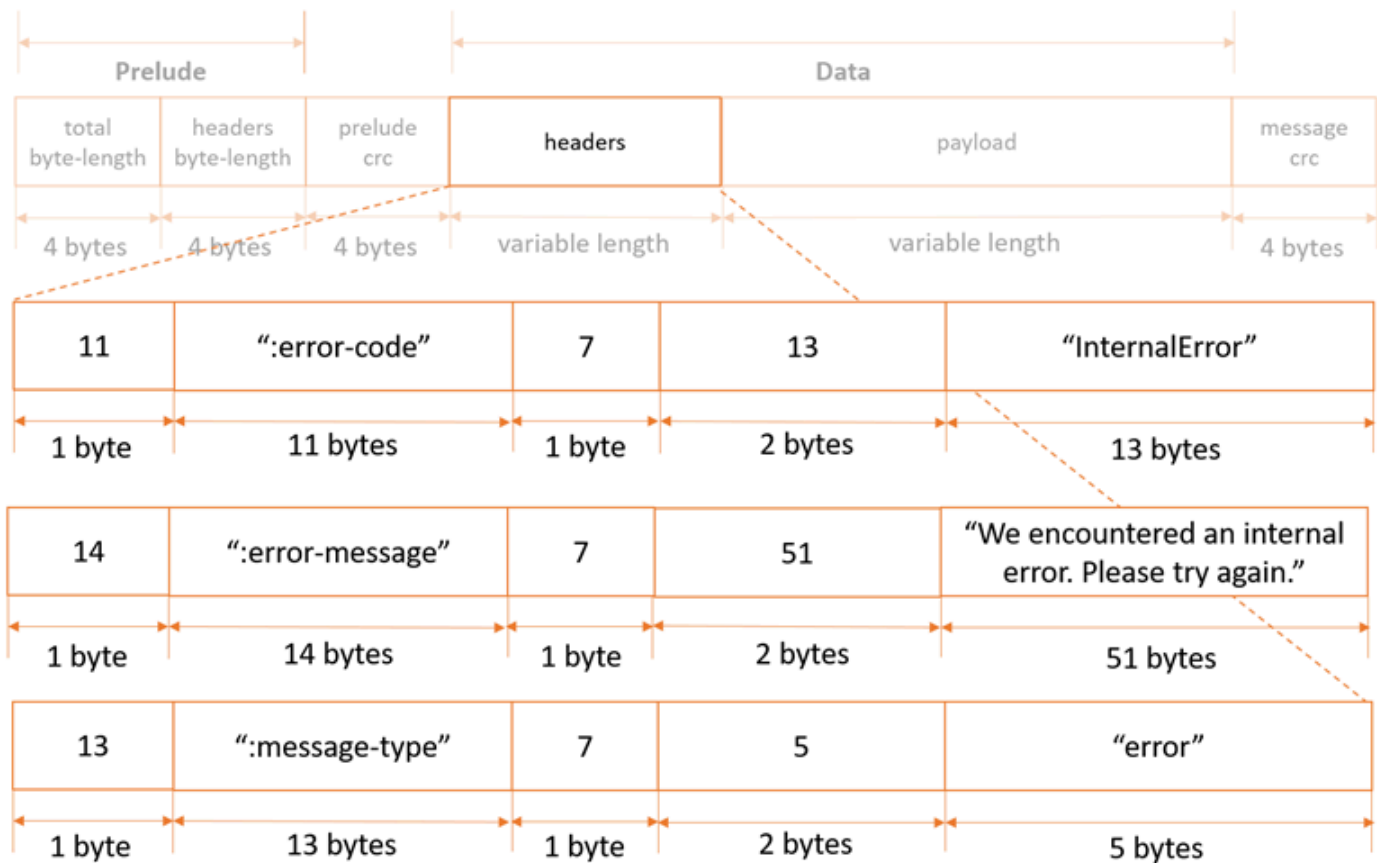
Spesifikasi muatan

Pesan akhir tidak memiliki muatan.

Pesan Kesalahan Tingkat Permintaan

Spesifikasi header

Request-level pesan kesalahan berisi tiga header, sebagai berikut:



Untuk daftar kemungkinan kode kesalahan dan pesan kesalahan, lihat [Daftar Kode Kesalahan Konten Objek SELECT](#).

Spesifikasi muatan

Request-level pesan kesalahan tidak memiliki payload.

Sumber Daya Terkait

- [SelectObjectContent](#)
- [GetObject](#)
- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)

Lampiran: objek OPTIONS

Deskripsi

Browser dapat mengirim permintaan preflight ini ke Amazon S3 untuk menentukan apakah dapat mengirim permintaan aktual dengan asal, metode HTTP, dan header tertentu.

Amazon S3 mendukung cross-origin resource sharing (CORS) dengan memungkinkan Anda menambahkan cors subresource pada bucket. Saat browser mengirimkan permintaan preflight ini, Amazon S3 merespons dengan mengevaluasi aturan yang ditentukan dalam cors konfigurasi.

Jika tidak cors diaktifkan pada bucket, maka Amazon S3 mengembalikan 403 Forbidden respons.

Untuk informasi selengkapnya tentang CORS, buka Meng [aktifkan Berbagi Cross-Origin Sumber Daya](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Permintaan

Sintaksis

```
OPTIONS /ObjectName HTTP/1.1
Host: BucketName.s3.amazonaws.com
Origin: Origin
Access-Control-Request-Method: HTTPMethod
Access-Control-Request-Headers: RequestHeader
```

Parameter Permintaan

Operasi ini tidak memperkenalkan parameter permintaan tertentu, tetapi mungkin berisi parameter permintaan apa pun yang diperlukan oleh permintaan aktual.

Header Permintaan

Nama	Deskripsi	Diperlukan
Origin	Mengidentifikasi asal permintaan lintas asal ke Amazon S3. Sebagai contoh, http://www.example.com.. Tipe: String	Ya

Nama	Deskripsi	Diperlukan
	Default: Tidak Ada	
Access-Control-Request-Method	Mengidentifikasi metode HTTP apa yang akan digunakan dalam permintaan aktual. Tipe: String Default: Tidak Ada	Ya
Access-Control-Request-Headers	Daftar header HTTP yang dipisahkan koma yang akan dikirim dalam permintaan aktual. Misalnya, untuk menempatkan objek dengan enkripsi sisi server, permintaan preflight ini akan menentukan apakah itu dapat menyertakan <code>x-amz-server-side-encryption</code> header dengan permintaan. Tipe: String Default: Tidak Ada	Tidak

Elemen permintaan

Implementasi operasi ini tidak menggunakan elemen permintaan.

Respons

Header Respons

Header	Deskripsi
Access-Control-Allow-Origin	Asal yang Anda kirim dalam permintaan Anda. Jika asal dalam permintaan Anda tidak diizinkan, Amazon S3 tidak akan menyertakan header ini dalam respons.

Header	Deskripsi
	Tipe: String
Access-Control-Max-Age	Berapa lama, dalam hitungan detik, hasil permintaan preflight dapat di-cache. Tipe: String
Access-Control-Allow-Methods	Metode HTTP yang dikirim dalam permintaan asli. Jika metode dalam permintaan tidak diizinkan, Amazon S3 tidak akan menyertakan header ini dalam respons. Tipe: String
Access-Control-Allow-Headers	Daftar header HTTP yang dibatasi koma yang dapat dikirim browser dalam permintaan yang sebenarnya. Jika salah satu header yang diminta tidak diizinkan, Amazon S3 tidak akan menyertakan header tersebut dalam respons, respon juga tidak akan berisi header dengan awalanAccess-Control . Tipe: String
Access-Control-Expose-Headers	Daftar header HTTP yang dipisahkan koma. Header ini menyediakan JavaScript klien dengan akses ke header ini dalam menanggapi permintaan yang sebenarnya. Tipe: String

Elemen Respons

Implementasi operasi ini tidak mengembalikan elemen respons.

Contoh

Contoh: Mengirim permintaan opsi preflight ke bucket yang diaktifkan **cors**

Browser dapat mengirim permintaan preflight ini ke Amazon S3 untuk menentukan apakah dapat mengirim permintaan PUT aktual dari `http://www.example.com` asal ke bucket Amazon S3 bernama `examplebucket`.

Permintaan Sampel

```
OPTIONS /exampleobject HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Origin: http://www.example.com
Access-Control-Request-Method: PUT
```

Contoh Respons

```
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: "1f1a1af1f1111111111111c11aed1da1"
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
```

Sumber Daya Terkait

- [GetBucketCors](#)
- [DeleteBucketCors](#)
- [PutBucketCors](#)

Lampiran: SOAP API

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

Bagian ini menjelaskan SOAP API sehubungan dengan operasi layanan, bucket, dan objek. Perhatikan bahwa permintaan SOAP, baik yang diautentikasi maupun anonim, harus dikirim ke Amazon S3 menggunakan SSL. Amazon S3 akan menampilkan pesan kesalahan saat Anda mengirim permintaan SOAP melalui HTTP.

Amazon S3 WSDL terbaru tersedia di docs.aws.amazon.com/2006-03-01/AmazonS3.wsdl.

Topik

- [Operasi pada Layanan \(SOAP API\)](#)
- [Operasi pada Bucket \(SOAP API\)](#)
- [Operasi pada Objek \(SOAP API\)](#)
- [Mengautentikasi permintaan SOAP](#)
- [Mengatur kebijakan akses dengan SOAP](#)
- [Elemen-elemen umum](#)
- [Tanggapan Kesalahan SOAP](#)

Operasi pada Layanan (SOAP API)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

Bagian ini menjelaskan operasi yang dapat Anda lakukan pada layanan Amazon S3.

Topik

- [ListAllMyBuckets \(API SABUN\)](#)

ListAllMyBuckets (API SABUN)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

ListAllMyBucketsOperasi mengembalikan daftar semua bucket yang dimiliki oleh pengirim permintaan.

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>
```

```
</Buckets>  
</ListAllMyBucketsResult>
```

Isi Respons

- **Owner :**

Ini memberikan informasi yang digunakan Amazon S3 untuk mewakili identitas Anda untuk tujuan otentikasi dan kontrol akses. ID adalah pengenalan unik dan permanen untuk pengembang yang membuat permintaan. DisplayName adalah nama yang dapat dibaca manusia yang mewakili pengembang yang membuat permintaan. Ini tidak unik, dan mungkin berubah seiring waktu. Kami menyarankan Anda mencocokkan nama Anda dengan DisplayName nama Forum Anda.

- **Name :**

Nama dari suatu bucket. Perhatikan bahwa jika salah satu bucket Anda baru saja dihapus, nama bucket yang dihapus mungkin masih ada dalam daftar ini untuk jangka waktu tertentu.

- **CreationDate :**

Waktu ember itu dibuat.

Kontrol Akses

Anda harus mengotentikasi dengan ID Kunci Ak AWS ses yang valid. Permintaan anonim tidak pernah diizinkan untuk membuat daftar bucket, dan Anda hanya dapat mencantumkan bucket yang Anda miliki.

Operasi pada Bucket (SOAP API)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

Bagian ini menjelaskan operasi yang dapat Anda lakukan pada bucket Amazon S3.

Topik

- [CreateBucket \(API SABUN\)](#)

- [DeleteBucket \(API SABUN\)](#)
- [ListBucket \(API SABUN\)](#)
- [GetBucketAccessControlPolicy \(API SABUN\)](#)
- [SetBucketAccessControlPolicy \(API SABUN\)](#)
- [GetBucketLoggingStatus \(API SABUN\)](#)
- [SetBucketLoggingStatus \(API SABUN\)](#)

CreateBucket (API SABUN)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

CreateBucketOperasi menciptakan ember. Tidak setiap string adalah nama bucket yang dapat diterima. Untuk informasi tentang batasan penamaan bucket, lihat [Berkas dengan Bucket Amazon S3](#).

Note

Untuk menentukan apakah nama bucket ada, gunakan ListBucket dan setel MaxKeys ke 0. Resp NoSuchBucket ons menunjukkan bahwa bucket tersedia, AccessDenied respons menunjukkan bahwa orang lain memiliki bucket, dan respons Sukses menunjukkan bahwa Anda memiliki bucket atau memiliki izin untuk mengaksesnya.

Example Buat bucket bernama “quote”

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>
```

Elemen

- **Bucket**: Nama bucket yang Anda coba buat.
- **AccessControlList**: Daftar kontrol akses untuk bucket baru. Elemen ini opsional. Jika tidak disediakan, bucket dibuat dengan kebijakan akses yang memberikan akses FULL_CONTROL kepada pemohon.

Kontrol Akses

Anda harus mengotentikasi dengan ID Kunci Ak AWS ses yang valid. Permintaan anonim tidak pernah diizinkan untuk membuat bucket.

Sumber Daya Terkait

- [ListBucket \(API SABUN\)](#)

DeleteBucket (API SABUN)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

DeleteBucketOperasi menghapus ember. Semua objek dalam bucket harus dihapus sebelum bucket itu sendiri dapat dihapus.

Example

Contoh ini menghapus bucket “tanda kutip”.

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>
```

Elemen

- **Bucket** : Nama bucket yang ingin Anda hapus.

Kontrol Akses

Hanya pemilik bucket yang diizinkan untuk menghapusnya, terlepas dari kebijakan kontrol akses pada bucket.

ListBucket (API SABUN)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

ListBucketOperasi mengembalikan informasi tentang beberapa item dalam bucket.

Untuk pengenalan umum operasi daftar, lihat Kunci [Objek Daftar](#).

Permintaan

Contoh ini mencantumkan hingga 1000 kunci dalam bucket “tanda kutip” yang memiliki awalan “catatan.”

Sintaks

```
<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>
```

Parameter

Nama	Deskripsi	Diperlukan
<code>prefix</code>	Membatasi respons ke kunci yang dimulai dengan awalan yang ditunjukkan. Anda dapat menggunakan awalan untuk memisahkan bucket menjadi kumpulan kunci yang berbeda dengan cara yang mirip dengan bagaimana sistem file menggunakan folder. Tipe: String Default: Tidak Ada	Tidak
<code>marker</code>	Menunjukkan di mana di bucket untuk memulai daftar. Daftar hanya akan menyertakan kunci yang terjadi secara leksikografis setelah penanda. Ini nyaman untuk paginasi: Untuk mendapatkan halaman hasil berikutnya gunakan kunci terakhir dari halaman saat ini sebagai penanda. Tipe: String	Tidak

Nama	Deskripsi	Diperlukan
	Default: Tidak Ada	
max-keys	Jumlah maksimum kunci yang ingin Anda lihat di badan respons. Server mungkin mengembalikan lebih sedikit dari banyak kunci ini, tetapi tidak akan mengembalikan lebih banyak. Tipe: String Default: Tidak Ada	Tidak
delimiter	Menyebabkan kunci yang berisi string yang sama antara awalan dan kejadian pertama pembatas digulung menjadi elemen hasil tunggal dalam koleksi. CommonPrefixes Kunci yang digulung ini tidak dikembalikan ke tempat lain dalam respons. Tipe: String Default: Tidak Ada	Tidak

Respon Sukses

Respons ini mengasumsikan bucket berisi kunci berikut:

```
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
```

Sintaksis

```
<?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>

```

Seperti yang Anda lihat, banyak bidang dalam respons menggunakan parameter permintaan. `IsTruncated`, `Contents`, dan `CommonPrefixes` merupakan satu-satunya elemen respons yang dapat berisi informasi baru.

Elemen Respons

Nama	Deskripsi
Contents	Metadata tentang setiap objek yang dikembalikan. Jenis: Metadata XML Leluhur: ListBucketResult
CommonPrefixes	Respons <code>CommonPrefixes</code> hanya dapat berisi jika Anda menentukan <code>delimiter</code>. Ketika Anda melakukannya, <code>CommonPrefixes</code> berisi semua (jika ada) kunci antara <code>Prefix</code> dan kejadian berikutnya dari string yang ditentukan oleh <code>delimiter</code>. Akibatnya, <code>CommonPrefixes</code> daftar kunci yang bertindak seperti subdirektori di direktori yang ditentukan oleh <code>Prefix</code>. Misalnya, jika <code>prefix</code> adalah <code>notes/</code> dan <code>delimiter</code> merupakan garis miring (<code>/</code>), in <code>notes/summer/july</code>, awalan umum adalah <code>notes/summer/</code>.

Nama	Deskripsi
	Tipe: String Leluhur: ListBucketResult
Delimiter	Menyebabkan kunci yang berisi string yang sama antara awalan dan kejadian pertama pembatas digulung menjadi elemen hasil tunggal dalam koleksi. CommonPrefixes Kunci yang digulung ini tidak dikembalikan ke tempat lain dalam respons. Tipe: String Leluhur: ListBucketResult
IsTruncated	Menentukan apakah (benar) atau tidak (salah) semua hasil dikembalikan. Semua hasil mungkin tidak dikembalikan jika jumlah hasil melebihi yang ditentukan oleh MaxKeys. Tipe: String Leluhur: boolean
Marker	Menunjukkan di mana di bucket untuk memulai daftar. Tipe: String Leluhur: ListBucketResult
MaxKeys	Jumlah maksimum kunci yang dikembalikan dalam badan respons. Tipe: String Leluhur: ListBucketResult
Name	Nama ember. Tipe: String Leluhur: ListBucketResult

Nama	Deskripsi
Prefix	Tombol yang dimulai dengan awalan yang ditunjukkan. Tipe: String Leluhur: ListBucketResult

Isi Respons

Untuk informasi tentang respons daftar, lihat Tanggapan [Kunci Listing](#).

Kontrol Akses

Untuk mencantumkan kunci bucket, Anda harus diberikan READ akses pada bucket.

GetBucketAccessControlPolicy (API SABUN)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

GetBucketAccessControlPolicyOperasi mengambil kebijakan kontrol akses untuk bucket.

Example

Contoh ini mengambil kebijakan kontrol akses untuk bucket “kutipan”.

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>a9a7b886d6f41bf9b1c61be666e9</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>
```

Isi Respons

Respons berisi kebijakan kontrol akses untuk bucket. Untuk penjelasan tanggapan ini, lihat Kebijakan Ak [ses SOAP](#).

Kontrol Akses

Anda harus memiliki READ_ACP hak ke bucket untuk mengambil kebijakan kontrol akses untuk bucket.

SetBucketAccessControlPolicy (API SABUN)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

SetBucketAccessControlPolicyOperasi menetapkan Kebijakan Kontrol Akses untuk bucket yang ada. Jika berhasil, Kebijakan Kontrol Akses sebelumnya untuk bucket sepenuhnya diganti dengan Kebijakan Kontrol Akses yang ditentukan.

Example

Berikan pengguna yang ditentukan (biasanya pemilik) FULL_CONTROL akses ke bucket “kutipan”.

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>
```

Kontrol Akses

Anda harus memiliki WRITE_ACP hak ke bucket untuk mengatur kebijakan kontrol akses untuk bucket.

GetBucketLoggingStatus (API SABUN)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

Meng `GetBucketLoggingStatus` ambil status logging untuk bucket yang ada.

Untuk pengenalan umum fitur ini, lihat [Log Server](#).

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>
```

Kontrol Akses

Hanya pemilik ember yang diizinkan untuk menjalankan operasi ini.

SetBucketLoggingStatus (API SABUN)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

SetBucketLoggingStatusOperasi memperbarui status logging untuk bucket yang ada.

Untuk pengenalan umum fitur ini, lihat [Log Server](#).

Example

Permintaan sampel ini memungkinkan logging akses server untuk bucket 'mybucket', dan mengonfigurasi log untuk dikirim ke 'mylogs' di bawah awalan '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>
```

Kontrol Akses

Hanya pemilik ember yang diizinkan untuk menjalankan operasi ini.

Operasi pada Objek (SOAP API)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

Bagian ini menjelaskan operasi yang dapat Anda lakukan pada objek Amazon S3.

Topik

- [PutObjectInline \(API SABUN\)](#)
- [PutObject \(API SABUN\)](#)
- [CopyObject \(API SABUN\)](#)
- [GetObject \(API SABUN\)](#)
- [GetObjectExtended \(API SABUN\)](#)
- [DeleteObject \(API SABUN\)](#)
- [GetObjectAccessControlPolicy \(API SABUN\)](#)
- [SetObjectAccessControlPolicy \(API SABUN\)](#)

PutObjectInline (API SABUN)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

PutObjectInlineOperasi menambahkan objek ke ember. Data untuk objek disediakan di badan pesan SOAP.

Jika objek sudah ada dalam bucket, objek baru akan menuliskannya karena Amazon S3 menyimpan permintaan tulis terakhir. Namun, Amazon S3 adalah sistem terdistribusi. Jika Amazon S3 menerima

beberapa permintaan tulis untuk objek yang sama hampir secara bersamaan, semua objek mungkin disimpan, meskipun hanya satu yang menang pada akhirnya. Amazon S3 tidak menyediakan penguncian objek; jika Anda membutuhkan ini, pastikan untuk memasukkannya ke dalam lapisan aplikasi Anda.

Untuk memastikan objek tidak rusak melalui jaringan, Anda dapat menghitung MD5 objek, MEMASUKkannya ke Amazon S3, dan membandingkan Etag yang dikembalikan dengan nilai MD5 yang dihitung.

PutObjectInline tidak cocok untuk digunakan dengan benda besar. Sistem membatasi operasi ini untuk bekerja dengan objek 1MB atau lebih kecil. PutObjectInline akan gagal dengan kode `InlineDataTooLargeError` status jika parameter Data mengkodekan objek yang lebih besar dari 1MB. Untuk mengunggah objek besar, pertimbangkan untuk menggunakan PutObject API non-inline, atau REST API sebagai gantinya.

Example

Contoh ini menulis beberapa teks dan metadata ke dalam objek "Nelson" di bucket "quote", memberikan pengguna (biasanya pemilik) FULL_CONTROL akses ke objek, dan membuat objek dapat dibaca oleh pihak anonim.

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>"828ef3fd96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2006-01-01T12:00:00.000Z</lastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>

```

Elemen

- **Bucket**: Ember untuk menambahkan objek.
- **Key**: Kunci untuk menetapkan ke objek.
- **Metadata**: Anda dapat memberikan pasangan metadata nama-nilai di elemen metadata. Ini akan disimpan dengan objek.
- **Data**: Bentuk data yang dikodekan basis 64.
- **ContentLength**: Panjang data dalam byte.
- **AccessControlList**: Daftar Kontrol Akses untuk sumber daya. Elemen ini opsional. Jika dihilangkan, pemohon diberikan FULL_CONTROL akses ke objek. Jika objek sudah ada, kebijakan kontrol akses yang sudah ada sebelumnya diganti.

Respons

- **ETag**: Tag entitas adalah hash MD5 dari objek yang dapat Anda gunakan untuk melakukan pengambilan bersyarat dari objek yang digunakan. `GetObjectExtended` ETag hanya mencerminkan perubahan pada konten objek, bukan metadatanya.
- **LastModified**: Stempel waktu Amazon S3 untuk objek yang disimpan.

Kontrol Akses

Anda harus memiliki **WRITE** akses ke ember untuk memasukkan benda ke dalam ember.

Sumber Daya Terkait

- [PutObject \(API SABUN\)](#)
- [CopyObject \(API SABUN\)](#)

PutObject (API SABUN)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

`PutObject` Operasi menambahkan objek ke ember. Data untuk objek dilampirkan sebagai lampiran DIME.

Untuk memastikan objek tidak rusak melalui jaringan, Anda dapat menghitung MD5 objek, MEMASUKkannya ke Amazon S3, dan membandingkan Etag yang dikembalikan dengan nilai MD5 yang dihitung.

Jika objek sudah ada dalam bucket, objek baru akan menuliskannya karena Amazon S3 menyimpan permintaan tulis terakhir. Namun, Amazon S3 adalah sistem terdistribusi. Jika Amazon S3 menerima beberapa permintaan tulis untuk objek yang sama hampir secara bersamaan, semua objek mungkin

disimpan, meskipun hanya satu yang menang pada akhirnya. Amazon S3 tidak menyediakan penguncian objek; jika Anda membutuhkan ini, pastikan untuk memasukkannya ke dalam lapisan aplikasi Anda.

Example

Contoh ini menempatkan beberapa data dan metadata dalam objek “Nelson” dari bucket “quote”, memberikan pengguna (biasanya pemilik) FULL_CONTROL akses ke objek, dan membuat objek dapat dibaca oleh pihak anonim. Dalam sampel ini, lampiran aktual tidak ditampilkan.

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>
  <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>
```

Elemen

- **Bucket**: Ember untuk menambahkan objek.
- **Key**: Kunci untuk menetapkan ke objek.
- **Metadata**: Anda dapat memberikan pasangan metadata nama-nilai di elemen metadata. Ini akan disimpan dengan objek.
- **ContentLength**: Panjang data dalam byte.
- **AccessControlList**: Daftar Kontrol Akses untuk sumber daya. Elemen ini opsional. Jika dihilangkan, pemohon diberikan FULL_CONTROL akses ke objek. Jika objek sudah ada, Kebijakan Kontrol Akses yang sudah ada sebelumnya diganti.

Respons

- **ETag**: Tag entitas adalah hash MD5 dari objek yang dapat Anda gunakan untuk melakukan pengambilan bersyarat dari objek yang digunakan. GetObjectExtended ETag hanya mencerminkan perubahan pada konten objek, bukan metadatanya.
- **LastModified**: Stempel waktu Amazon S3 untuk objek yang disimpan.

Kontrol Akses

Untuk memasukkan benda ke dalam ember, Anda harus memiliki WRITE akses ke ember.

Sumber Daya Terkait

- [CopyObject \(API SABUN\)](#)

CopyObject (API SABUN)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

Deskripsi

CopyObjectOperasi membuat salinan objek ketika Anda menentukan kunci dan bucket dari objek sumber dan kunci dan bucket tujuan target.

Saat menyalin objek, Anda dapat mempertahankan semua metadata (default) atau menentukan metadata baru. Namun, ACL tidak dipertahankan dan diatur `private` untuk pengguna yang membuat permintaan. Untuk mengganti setelan ACL default, tentukan ACL baru saat membuat permintaan salinan. Untuk informasi lebih lanjut, lihat [Menggunakan ACL](#).

Semua permintaan salinan harus diautentikasi. Selain itu, Anda harus memiliki akses baca ke objek sumber dan akses menulis ke bucket tujuan. Untuk informasi selengkapnya, lihat [Menggunakan Akses Auth](#).

Untuk hanya menyalin objek dalam kondisi tertentu, seperti apakah Etag cocok atau apakah objek dimodifikasi sebelum atau setelah tanggal tertentu, gunakan parameter permintaan `CopySourceIfUnmodifiedSince`, `CopyIfUnmodifiedSince`, `CopySourceIfMatch`, atau `CopySourceIfNoneMatch`.

Note

Anda mungkin perlu mengonfigurasi batas waktu soket tumpukan SOAP untuk menyalin objek besar.

Sintaksis Permintaan

```
<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>

```

Parameter Permintaan

Nama	Deskripsi	Diperlukan
SourceBucket	Nama bucket sumber. Tipe: String Default: Tidak Ada Kendala: Bucket sumber yang valid.	Ya
SourceKey	Nama kunci dari objek sumber. Tipe: String	Ya

Nama	Deskripsi	Diperlukan
	Default: Tidak Ada Batasan: Kunci untuk objek sumber yang valid yang memiliki akses BACA.	
DestinationBucket	Nama bucket tujuan. Tipe: String Default: Tidak Ada Kendala: Anda harus memiliki akses WRITE ke bucket tujuan.	Ya
DestinationKey	Kunci objek tujuan. Tipe: String Default: Tidak Ada Kendala: Anda harus memiliki akses WRITE ke bucket tujuan.	Ya

Nama	Deskripsi	Diperlukan
MetadataDirective	Menentukan apakah metadata disalin dari objek sumber atau diganti dengan metadata yang disediakan dalam permintaan. Tipe: String Default: COPY Nilai yang valid: COPY REPLACE Kendala: Nilai selain COPY atau REPLACE akan mengakibatkan kesalahan langsung. Anda tidak dapat menyalin objek ke dirinya sendiri kecuali MetadataDirective header ditentukan dan nilainya ditetapkan keREPLACE.	Tidak
Metadata	Menentukan pasangan nama-nilai metadata untuk disetel untuk objek. Jika disetel keCOPY, MetadataDirective semua metadata diabaikan. Tipe: String Default: Tidak Ada Kendala: Tidak ada.	Tidak
AccessControlList	Memberikan akses ke pengguna melalui alamat email atau ID pengguna kanonik. Tipe: String Default: Tidak ada Batasan: Tidak Ada	Tidak

Nama	Deskripsi	Diperlukan
CopySourceIfMatch	Menyalin objek jika tag entitasnya (ETag) cocok dengan tag yang ditentukan; jika tidak, kembalikan a PreconditionFailed. Tipe: String Default: Tidak Ada Kendala: Tidak ada. Jika Etag tidak cocok, objek tidak disalin.	Tidak
CopySourceIfNoneMatch	Menyalin objek jika tag entitasnya (ETag) berbeda dari Etag yang ditentukan; jika tidak mengembalikan kesalahan. Tipe: String Default: Tidak Ada Kendala: Tidak ada.	Tidak
CopySourceIfUnmodifiedSince	Menyalin objek jika belum dimodifikasi sejak waktu yang ditentukan; jika tidak mengembalikan a PreconditionFailed. Jenis: DateTime Default: Tidak Ada	Tidak
CopySourceIfModifiedSince	Menyalin objek jika telah dimodifikasi sejak waktu yang ditentukan; jika tidak mengembalikan kesalahan. Jenis: DateTime Default: Tidak Ada	Tidak

Sintaksis Respons

```
<CopyObjectResponse xmlns="http://bucket_name.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"etag"</ETag>
    <LastModified>timestamp</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

Elemen Respons

Berikut ini adalah daftar elemen respons.

Note

API SOAP tidak mengembalikan spasi putih tambahan. Spasi tambahan hanya dikembalikan oleh REST API.

Nama	Deskripsi
Etag	Mengembalikan etag objek baru. ETag hanya mencerminkan perubahan pada konten objek, bukan metadatanya. Tipe: String Leluhur: CopyObjectResult
LastModified	Mengembalikan tanggal objek terakhir diubah. Tipe: String Leluhur: CopyObjectResult

Untuk informasi tentang elemen respons umum, lihat [Menggunakan Header Respon Kesalahan REST](#).

Kesalahan Khusus

Tidak ada kesalahan khusus untuk operasi ini. Untuk informasi tentang kesalahan Amazon S3 umum, lihat [Daftar kode kesalahan](#).

Contoh

Contoh ini menyalin objek `flotsam` dari bucket `pacific` ke objek `jetsam` dari bucket `atlantic`, dengan mempertahankan metadatanya.

Permintaan Sampel

```
<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>
```

Contoh Respons

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

Contoh ini menyalin objek `"tweedledee"` dari bucket `wonderland` ke objek `"tweedledum"` dari bucket `wonderland`, menggantikan metadatanya.

Permintaan Sampel

```
<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>
  <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>
```

Contoh Respons

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fd96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

Sumber Daya Terkait

- [PutObject \(API SABUN\)](#)
- [PutObjectInline \(API SABUN\)](#)

GetObject (API SABUN)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

GetObjectOperasi mengembalikan versi objek saat ini. Jika Anda mencoba ke GetObject objek yang memiliki penanda hapus sebagai versi saat ini, S3 mengembalikan kesalahan 404. Anda tidak dapat menggunakan SOAP API untuk mengambil versi objek tertentu. Untuk melakukannya, gunakan REST API. Untuk informasi selengkapnya, lihat Pem [buatan versi](#). Untuk opsi lainnya, gunakan [GetObjectExtended \(API SABUN\)](#) operasi.

 Note

Nama kunci objek dengan nilai “soap” tidak didukung untuk [permintaan bergaya yang dihosting virtual](#). Untuk nilai nama kunci objek di mana “soap” digunakan, [URL bergaya jalur](#) harus digunakan sebagai gantinya.

Example

Contoh ini mendapatkan objek “Nelson” dari bucket “quote”.

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>
<ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
</GetObjectResponse>
</GetObjectResponse>
```

Elemen

- **Bucket**: Ember dari mana untuk mengambil objek.
- **Key**: Kunci yang mengidentifikasi objek.
- **GetMetadata**: Metadata dikembalikan dengan objek jika ini benar.
- **GetData**: Data objek dikembalikan jika ini benar.
- **InlineData**: Jika ini benar, maka data dikembalikan, basis 64 dikodekan, sebagai bagian dari badan SOAP dari respons. Jika salah, maka data dikembalikan sebagai lampiran SOAP. InlineData Opsi ini tidak cocok untuk digunakan dengan benda besar. Sistem membatasi operasi ini untuk bekerja dengan 1MB data atau kurang. Per GetObject mintaan dengan set InlineData bendera akan gagal dengan kode `InlineDataTooLargeError` status jika parameter Data yang dihasilkan akan dikodekan lebih dari 1MB. Untuk mengunduh objek besar, pertimbangkan untuk memanggil GetObject tanpa menyet InlineData el flag, atau gunakan REST API sebagai gantinya.

Elemen yang dikembalikan

- **Metadata**: Metadata berpasangan nama-nilai disimpan dengan objek.
- **Data**: Jika InlineData benar dalam permintaan, ini berisi data objek yang dikodekan basis 64.
- **LastModified**: Waktu objek disimpan di Amazon S3.
- **ETag**: Tag entitas objek. Ini adalah hash dari objek yang dapat digunakan untuk melakukan perolehan bersyarat. ETag hanya mencerminkan perubahan pada konten objek, bukan metadatanya.

Kontrol Akses

Anda dapat membaca objek hanya jika Anda telah diberikan READ akses ke objek.

UNDUHAN SOAP Terpotong dan Dapat Dilanjutkan

Untuk memberikan GET fleksibilitas, Amazon S3 mendukung unduhan yang dipotong dan dapat dilanjutkan.

Pilih dari yang berikut ini:

- Untuk unduhan objek besar, Anda mungkin ingin memecahnya menjadi potongan-potongan yang lebih kecil. Untuk informasi selengkapnya, lihat [Rentang GET](#)
- Untuk GET operasi yang gagal, Anda dapat merancang aplikasi Anda untuk mengunduh sisanya, bukan seluruh file. Untuk informasi selengkapnya, lihat [REST GET Pemulihan Kesalahan](#)

Rentang GET

Untuk beberapa klien, Anda mungkin ingin memecah unduhan besar menjadi unduhan yang lebih kecil. Untuk memecah GET menjadi unit yang lebih kecil, gunakan Range.

Sebelum Anda dapat memecah GET menjadi unit yang lebih kecil, Anda harus menentukan ukurannya. Misalnya, permintaan berikut mendapatkan ukuran objek 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 mengembalikan respons berikut.

```
<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>
```



```
</Contents>
</ListBucketResult>
```

Berikut ini adalah permintaan yang mengunduh megabyte pertama dari objek 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 mengembalikan megabyte pertama dari file dan Etag 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>
```

Untuk memastikan file tidak berubah sejak bagian sebelumnya diunduh, tentukan IfMatch elemen. Meskipun IfMatch elemen tidak diperlukan, disarankan untuk konten yang cenderung berubah.

Berikut ini adalah permintaan yang mendapatkan sisa file, menggunakan header IfMatch permintaan.

```
<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>"828ef3fd96f00ad9f27c383fc9ac7f"</IfMatch>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Amazon S3 mengembalikan respons berikut dan sisa 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>"828ef3fd96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>
```

Berversi GetObject

Permintaan berikut mengembalikan versi objek yang ditentukan dalam 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>

```

Contoh Respons

```

<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>

```

REST GET Pemulihan Kesalahan

Jika objek GET gagal, Anda bisa mendapatkan sisa file dengan menentukan rentang untuk diunduh. Untuk melakukannya, Anda harus mendapatkan ukuran objek menggunakan `ListBucket` dan melakukan rentang GET pada sisa file. Untuk informasi selengkapnya, lihat [GetObjectExtended \(API SABUN\)](#).

Sumber Daya Terkait

[Operasi pada Objek \(SOAP API\)](#)

GetObjectExtended (API SABUN)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

GetObjectExtended persis seperti [GetObject \(API SABUN\)](#), kecuali bahwa ia mendukung elemen tambahan berikut yang dapat digunakan untuk mencapai banyak fungsi yang sama yang disediakan oleh header GET HTTP (pergi ke <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>).

GetObjectExtended mendukung elemen-elemen berikut selain yang didukung oleh GetObject:

- **ByteRangeStart, ByteRangeEnd**: Elemen-elemen ini menentukan bahwa hanya sebagian dari data objek yang harus diambil. Mereka mengikuti perilaku rentang byte HTTP (pergi ke [http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html #sec14.35](http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35)).
- **IfModifiedSince**: Kembalikan objek hanya jika stempel waktu objek lebih lambat dari stempel waktu yang ditentukan. (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.25>)
- **IfUnmodifiedSince**: Kembalikan objek hanya jika stempel waktu objek lebih awal dari atau sama dengan stempel waktu yang ditentukan. (buka [http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html #sec14.28](http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.28))
- **IfMatch**: Kembalikan objek hanya jika ETag-nya cocok dengan tag yang disediakan. (buka [http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html #sec14.24](http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.24))
- **IfNoneMatch**: Kembalikan objek hanya jika ETag-nya tidak cocok dengan tag yang disediakan. (buka [http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html #sec14.26](http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.26))
- **ReturnCompleteObjectOnConditionFailure**: ReturnCompleteObjectOnConditionFailure: Jika benar, maka jika permintaan menyertakan elemen rentang dan satu atau kedua IfUnmodifiedSince/IfMatch elemen, dan kondisinya gagal, kembalikan seluruh objek daripada kesalahan. Ini memungkinkan If-Range fungsionalitas (buka [http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html #sec14.27](http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.27)).

DeleteObject (API SABUN)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

DeleteObjectOperasi menghapus objek yang ditentukan dari Amazon S3. Setelah dihapus, tidak ada metode untuk mengembalikan atau membatalkan penghapusan objek.

Note

Jika Anda menghapus objek yang tidak ada, Amazon S3 akan mengembalikan hasil (bukan pesan kesalahan).

Example

Contoh ini menghapus objek "Nelson" dari bucket "quote".

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>
```

Elemen

- Bucket : Ember yang menampung objek.
- Key : Kunci yang mengidentifikasi objek.

Kontrol Akses

Anda dapat menghapus objek hanya jika Anda memiliki WRITE akses ke bucket, terlepas dari siapa yang memiliki objek atau hak apa yang diberikan padanya.

GetObjectAccessControlPolicy (API SABUN)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

GetObjectAccessControlPolicyOperasi mengambil kebijakan kontrol akses untuk objek.

Example

Contoh ini mengambil kebijakan kontrol akses untuk objek “Nelson” dari bucket “quote”.

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>
```

Isi Respons

Respons berisi kebijakan kontrol akses untuk bucket. Untuk penjelasan tanggapan ini, SO [AP Access Policy](#).

Kontrol Akses

Anda harus memiliki READ_ACP hak atas objek untuk mengambil kebijakan kontrol akses untuk suatu objek.

SetObjectAccessControlPolicy (API SABUN)

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

SetObjectAccessControlPolicyOperasi menetapkan kebijakan kontrol akses untuk objek yang ada. Jika berhasil, kebijakan kontrol akses sebelumnya untuk objek sepenuhnya diganti dengan kebijakan kontrol akses yang ditentukan.

Example

Contoh ini memberikan pengguna tertentu (biasanya pemilik) FULL_CONTROL akses ke objek "Nelson" dari bucket "quote".

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>
```

Key

Kontrol Akses

Anda harus memiliki WRITE_ACP hak atas objek untuk mengatur kebijakan kontrol akses untuk bucket.

Mengautentikasi permintaan SOAP

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

Setiap permintaan non-anonim harus berisi informasi autentikasi untuk menetapkan identitas pengguna utama yang membuat permintaan tersebut. Di dalam SOAP, informasi autentikasi dimasukkan ke dalam elemen-elemen permintaan SOAP berikut:

- ID K AWS unci Akses Anda

Note

Saat membuat permintaan SOAP yang diautentikasi, kredensial keamanan sementara tidak didukung. Untuk informasi selengkapnya tentang jenis-jenis kredensial, lihat [Membuat permintaan](#).

- **Timestamp**: Ini harus berupa DateTime (buka <http://www.w3.org/TR/xmlschema-2/#dateTime>) di zona waktu Waktu Universal Terkoordinasi (Greenwich Mean Time), seperti. 2009-01-01T12:00:00.000Z Otorisasi akan gagal jika stempel waktu ini lebih dari 15 menit dari jam di server Amazon S3.
- **Signature**: Ringkasan RFC 2104 (HMAC-SHA1 buka <http://www.ietf.org/rfc/rfc2104.txt>) dari penggabungan "Amazons3" + OPERATION+Timestamp, menggunakan Kunci Akses Rahasia Anda sebagai kuncinya. AWS Misalnya, dalam permintaan CreateBucket sampel berikut, elemen tanda tangan akan berisi HMAC-SHA1 ringkasan nilai "AmazonS3CreateBucket2009-01-01T12:00:00.000Z":

Misalnya, dalam permintaan CreateBucket sampel berikut, elemen tanda tangan akan berisi HMAC-SHA1 ringkasan nilai "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

Permintaan SOAP, baik yang diautentikasi maupun yang anonim, harus dikirim ke Amazon S3 menggunakan SSL. Amazon S3 akan menampilkan pesan kesalahan saat Anda mengirim permintaan SOAP melalui HTTP.

Important

Karena terdapat perbedaan interpretasi tentang cara menempatkan presisi waktu ekstra, pengguna .NET harus berhati-hati agar tidak mengirimkan stempel waktu Amazon S3 yang terlalu spesifik. Ini dapat dicapai dengan menyusun DateTime objek secara manual dengan presisi hanya dalam milidetik.

Mengatur kebijakan akses dengan SOAP

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

Kontrol akses dapat diatur pada saat bucket atau objek ditulis dengan memasukkan elemen "AccessControlList" dengan permintaan keCreateBucket,PutObjectInline, atauPutObject. AccessControlList Elemen ini dijelaskan dalam [Identitas dan Manajemen Akses untuk Amazon S3](#). Jika tidak ada daftar kontrol akses yang ditentukan dengan operasi ini, sumber daya dibuat dengan kebijakan akses default yang memberikan akses FULL_CONTROL pemohon (ini terjadi bahkan jika permintaan tersebut adalah PutObjectInline atau PutObject permintaan untuk objek yang sudah ada).

Berikut ini adalah permintaan yang menulis data ke sebuah objek, membuat objek tersebut dapat dibaca oleh pengguna utama anonim, dan memberi pengguna tersebut hak-hak FULL_CONTROL ke bucket (Sebagian besar pengembang akan memberikan akses FULL_CONTROL ke bucket mereka sendiri).

Example

Berikut ini adalah permintaan yang menulis data ke sebuah objek dan membuat objek tersebut dapat dibaca oleh pengguna utama anonim.

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>"828ef3fd96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2009-01-01T12:00:00.000Z</LastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>
```

Kebijakan kontrol akses dapat dibaca atau diatur untuk bucket atau objek yang sudah ada dengan menggunakan metode `GetBucketAccessControlPolicy`, `GetObjectAccessControlPolicy`, `SetBucketAccessControlPolicy`, dan `SetObjectAccessControlPolicy`. Untuk informasi lebih lanjut, lihat penjelasan terperinci tentang metode-metode tersebut.

Elemen-elemen umum

Anda dapat menyertakan elemen-elemen terkait otorisasi berikut dengan permintaan SOAP:

- `AWSAccessKeyId`: ID AWS Kunci Akses dari pemohon
- `Timestamp`: Waktu saat ini pada sistem Anda
- `Signature`: Tanda tangan untuk permintaan

Tanggapan Kesalahan SOAP

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

Dalam SOAP, hasil kesalahan dikembalikan kepada klien sebagai kesalahan SOAP, dengan kode respons HTTP 500. Jika Anda tidak menerima kesalahan SOAP, maka permintaan Anda berhasil. Kode kesalahan Amazon S3 SOAP terdiri dari kode kesalahan SOAP 1.1 standar (baik "Server" atau "Klien") yang digabungkan dengan kode kesalahan Amazon S3-specific. Misalnya: "Server.InternalError" atau "Client.NoSuchBucket". Elemen string kesalahan SOAP berisi pesan kesalahan umum yang dapat dibaca oleh manusia dalam bahasa Inggris. Terakhir, elemen detail kesalahan SOAP mengandung berbagai informasi yang relevan dengan kesalahan.

Misalnya, jika Anda mencoba menghapus objek “Fred”, yang tidak ada, badan respons SOAP berisi kesalahan “NoSuchKey” SOAP.

Contoh berikut menunjukkan contoh respons kesalahan 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>
```

Tabel berikut menjelaskan elemen respons kesalahan SOAP

Nama	Deskripsi
Detail	Wadah untuk kunci yang terlibat dalam kesalahan Tipe: Kontainer Leluhur: Body.Fault
Fault	Wadah untuk informasi kesalahan. Tipe: Kontainer Leluhur: Tubuh
Faultcode	Kode kesalahan adalah string yang secara unik mengidentifikasi kondisi kesalahan. Ini dimaksudkan untuk dibaca dan dipahami oleh program yang mendeteksi dan menangani kesalahan berdasarkan tipe. Untuk informasi selengkapnya, lihat Daftar Kode Kesalahan . Tipe: String Leluhur: Body.Fault
Faultstring	String kesalahan berisi deskripsi umum dari kondisi kesalahan dalam bahasa Inggris. Ini ditujukan untuk audiens manusia. Program sederhana menampilkan

Nama	Deskripsi
	an pesan secara langsung kepada pengguna akhir jika mereka mengalami kondisi kesalahan yang tidak mereka ketahui atau tidak peduli untuk menanganinya. Program canggih dengan penanganan kesalahan yang lebih lengkap dan internasionalisasi yang tepat lebih cenderung mengabaikan string kesalahan. Tipe: String Leluhur: Body.Fault
Key	Mengidentifikasi kunci yang terlibat dalam kesalahan Tipe: String Leluhur: Body.Fault

Lampiran: Permintaan otentikasi (AWS tanda tangan versi 2)

Important

Bagian ini menjelaskan cara mengotentikasi permintaan menggunakan Tanda T AWS angan Versi 2. Tanda Tangan Versi 2 akan dinonaktifkan (dihilangkan), Amazon S3 hanya akan menerima permintaan API yang ditandatangani menggunakan Tanda Tangan Versi 4. Untuk informasi selengkapnya, lihat [AWS Signature Versi 2 dimatikan \(tidak digunakan lagi\) untuk Amazon S3](#)

Signature Version 4 didukung di semua Wilayah AWS, dan ini adalah satu-satunya versi yang didukung untuk Wilayah baru. Untuk informasi selengkapnya, lihat Men [gotentikasi Permintaan \(AWS Tanda Tangan Versi 4\)](#) di Referensi API Layanan Penyimpanan Sederhana Amazon.

Amazon S3 menawarkan kemampuan mengidentifikasi versi tanda tangan API yang digunakan untuk menandatangani permintaan. Penting untuk mengidentifikasi apakah salah satu alur kerja Anda menggunakan penandatanganan dengan Tanda Tangan Versi 2 dan meningkatkannya menjadi Tanda Tangan Versi 4 untuk mencegah dampaknya terhadap bisnis Anda.

- Jika Anda menggunakan log CloudTrail peristiwa (opsi yang disarankan), lihat [Mengidentifikasi permintaan Amazon S3 Signature Versi 2 dengan menggunakan CloudTrail](#) cara menanyakan dan mengidentifikasi permintaan tersebut.
- Jika Anda menggunakan log Akses Server Amazon S3, lihat [Menggunakan log akses server Amazon S3 untuk mengidentifikasi permintaan](#)

Topik

- [Mengotentikasi permintaan menggunakan REST API \(AWS tanda tangan versi 2\)](#)
- [Menandatangani dan mengotentikasi permintaan REST \(AWS tanda tangan versi 2\)](#)
- [Browser-based unggah menggunakan POST \(AWS tanda tangan versi 2\)](#)

Mengotentikasi permintaan menggunakan REST API (AWS tanda tangan versi 2)

Saat mengakses Amazon S3 menggunakan REST, Anda harus menyediakan item-item berikut dalam permintaan Anda agar permintaan tersebut dapat diautentikasi:

Elemen permintaan

- **AWS ID kunci akses** — Setiap permintaan harus berisi ID kunci akses dari identitas yang Anda gunakan untuk mengirim permintaan Anda.
- **Tanda Tangan**—Setiap permintaan harus berisi tanda tangan permintaan yang valid, jika tidak, permintaan akan ditolak.

Tanda tangan permintaan dihitung menggunakan kunci akses rahasia, yang merupakan rahasia bersama yang hanya diketahui oleh Anda dan AWS.

- **Stempel waktu**—Setiap permintaan harus berisi tanggal dan waktu pembuatan permintaan yang direpresentasikan sebagai string dalam UTC.
- **Tanggal**—Setiap permintaan harus mencantumkan stempel waktu permintaan.

Tergantung pada tindakan API yang digunakan, Anda bisa memberikan tanggal dan waktu kedaluwarsa untuk permintaan, sebagai pengganti atau sebagai tambahan dari stempel waktu. Lihat topik autentikasi untuk tindakan tertentu agar dapat menentukan hal yang diperlukan.

Berikut adalah langkah umum untuk mengautentikasi permintaan ke Amazon S3. Anggaplah bahwa Anda telah memiliki kredensial keamanan, ID kunci akses, dan kunci akses rahasia yang diperlukan.

You**1 Create a request:****Request**

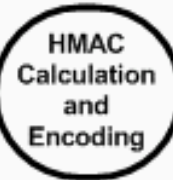
AccessKeyId = ...
Action = ...
Timestamp = ...
ParameterA = ...

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

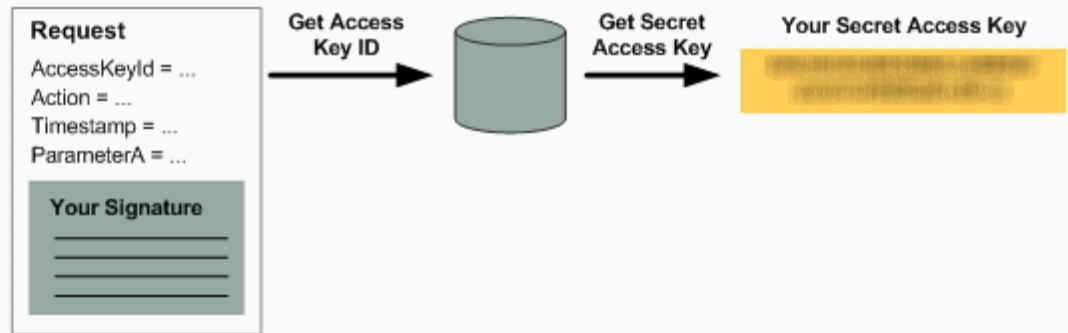


AWS

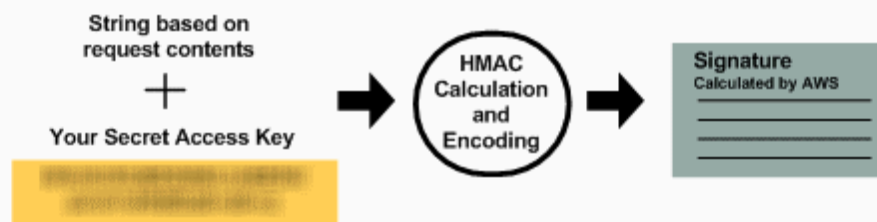
- 1 Buat permintaan ke AWS.
- 2 Hitung tanda tangan menggunakan kunci akses rahasia Anda.
- 3 Kirim permintaan ke Amazon S3. Sertakan ID kunci akses dan tanda tangan dalam permintaan Anda. Amazon S3 melakukan tiga langkah berikutnya.

AWS

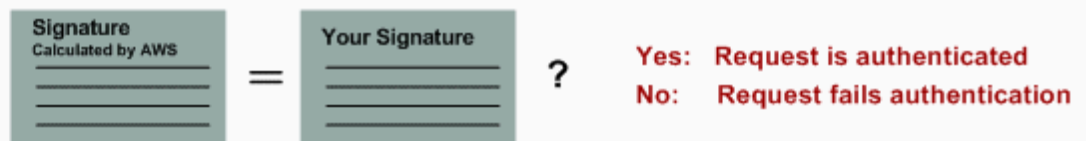
4 Retrieve your Secret Access Key:



5 Create an HMAC-SHA1 signature:



6 Compare the two signatures:



4 Amazon S3 menggunakan ID kunci akses untuk mencari kunci akses rahasia Anda.

5 Amazon S3 menghitung tanda tangan dari data permintaan dan kunci akses rahasia menggunakan algoritma yang sama dengan yang Anda gunakan untuk menghitung tanda tangan yang dikirim dalam permintaan.

6 Jika tanda tangan yang dihasilkan oleh Amazon S3 sesuai dengan yang Anda kirim dalam permintaan, permintaan tersebut dianggap benar. Jika perbandingan gagal, permintaan dibatalkan, dan Amazon S3 akan memberikan respons kesalahan.

Informasi autentikasi terperinci

Untuk informasi terperinci tentang autentikasi REST, lihat [Menandatangani dan mengotentikasi permintaan REST \(AWS tanda tangan versi 2\)](#).

Menandatangani dan mengotentikasi permintaan REST (AWS tanda tangan versi 2)

Topik

- [Menggunakan kredensial keamanan sementara](#)
- [Header autentikasi](#)
- [Kanonikalisasi permintaan untuk penandatanganan](#)
- [Membangun CanonicalizedResource elemen](#)
- [Membangun CanonicalizedAmzHeaders elemen](#)
- [Elemen header StringToSign HTTP posisi versus bernama](#)
- [Ketentuan stempel waktu](#)
- [Contoh-contoh autentikasi](#)
- [Masalah penandatanganan permintaan REST](#)
- [Alternatif autentikasi permintaan string kueri](#)

Note

Topik ini menjelaskan tentang autentikasi permintaan dengan menggunakan Tanda Tangan Versi 2. Amazon S3 kini mendukung Tanda Tangan Versi 4 terbaru. Versi tanda tangan terbaru ini didukung di semua wilayah dan setiap wilayah baru setelah 30 Januari 2014 hanya akan mendukung Tanda Tangan Versi 4. Untuk informasi selengkapnya, buka [Permintaan Otentikasi \(Tanda Tangan AWS Versi 4\)](#) di Referensi API Layanan Penyimpanan Sederhana Amazon.

Autentikasi adalah proses pembuktian identitas Anda pada sistem. Identitas adalah faktor penting dalam keputusan kontrol akses Amazon S3. Permintaan diizinkan atau ditolak sebagian berdasarkan identitas pemohon. Sebagai contoh, hak untuk membuat bucket disediakan untuk pengembang yang terdaftar dan (secara default) hak untuk membuat objek di dalam bucket disediakan untuk

pemilik bucket yang bersangkutan. Sebagai pengembang, Anda akan membuat permintaan yang menginvokasi hak istimewa ini, jadi Anda perlu membuktikan identitas pada sistem dengan melakukan autentikasi atas permintaan Anda. Dalam bagian ini, akan ditunjukkan caranya.

 Note

Konten dalam bagian ini tidak berlaku untuk POST HTTP. Untuk informasi selengkapnya, lihat [Browser-based unggah menggunakan POST \(AWS tanda tangan versi 2\)](#).

API REST Amazon S3 menggunakan skema HTTP kustom berdasarkan kunci-HMAC (Kode Autentikasi Pesan Hash) untuk autentikasi. Untuk mengautentikasi permintaan, pertama-tama Anda perlu menggabungkan elemen-elemen yang dipilih dari permintaan tersebut untuk membentuk string. Anda kemudian menggunakan kunci akses AWS rahasia Anda untuk menghitung HMAC dari string itu. Secara informal, proses ini disebut “menandatangani permintaan,” dan output dari algoritma HMAC adalah tanda tangan, karena mensimulasikan properti keamanan dari tanda tangan nyata. Terakhir, Anda perlu menambahkan tanda tangan ini sebagai parameter permintaan dengan menggunakan sintaks yang dijelaskan dalam bagian ini.

Ketika sistem menerima permintaan yang diautentikasi, ia mengambil kunci akses AWS rahasia yang Anda klaim miliki dan menggunakannya dengan cara yang sama untuk menghitung tanda tangan untuk pesan yang diterimanya. Kemudian, sistem akan membandingkan tanda tangan yang dihitung dengan tanda tangan yang diberikan oleh pemohon. Jika kedua tanda tangan cocok, sistem menyimpulkan bahwa pemohon harus memiliki akses ke kunci akses AWS rahasia dan karena itu bertindak dengan otoritas kepala sekolah kepada siapa kunci dikeluarkan. Jika kedua tanda tangan tidak cocok, permintaan akan dibatalkan dan sistem merespons dengan mengirimkan pesan kesalahan.

Example Permintaan REST Amazon S3 yang telah diautentikasi

```
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=
```

Menggunakan kredensial keamanan sementara

Jika Anda menandatangani permintaan menggunakan kredensial keamanan sementara (lihat [Membuat permintaan](#)), Anda harus menyertakan token keamanan yang sesuai dalam permintaan dengan menambahkan header `x-amz-security-token`.

Saat Anda memperoleh kredensial keamanan sementara menggunakan API AWS Security Token Service, respons tersebut mencakup kredensial keamanan sementara dan token sesi. Anda memberikan nilai token sesi dalam header `x-amz-security-token` saat Anda mengirim permintaan ke Amazon S3. Untuk informasi tentang AWS Security Token Service API yang disediakan oleh IAM, buka [Tindakan](#) di Panduan Refer AWS Security Token Service API.

Header autentikasi

API REST Amazon S3 menggunakan header HTTP `Authorization` standar untuk memberikan informasi autentikasi. (Nama header standar tidak disarankan karena memuat informasi autentikasi, bukan otorisasi.) Berdasarkan skema autentikasi Amazon S3, header Otorisasi memiliki bentuk berikut ini:

```
Authorization: AWS AWSAccessKeyId:Signature
```

Pengembang diberikan ID kunci AWS akses dan kunci akses AWS rahasia ketika mereka mendaftar. Untuk autentikasi permintaan, elemen `AWSAccessKeyId` mengidentifikasi ID kunci akses yang digunakan untuk menghitung tanda tangan dan secara tidak langsung, juga mengidentifikasi pengembang yang mengajukan permintaan.

`Signature` Elemen adalah RFC 2104 HMAC-SHA1 dari elemen yang dipilih dari permintaan, sehingga `Signature` bagian dari header Otorisasi akan bervariasi dari permintaan ke permintaan. Jika tanda tangan permintaan yang dihitung oleh sistem cocok dengan yang diser `Signature` takan dengan permintaan, pemohon akan menunjukkan kepemilikan kunci akses AWS rahasia. Permintaan tersebut kemudian akan diproses menggunakan identitas tersebut dan dengan otoritas dari pengembang yang diberikan kunci tersebut.

Berikut ini adalah pseudogrammar yang mengilustrasikan konstruksi header permintaan `Authorization`. (Dalam contoh tersebut, `\n` berarti titik kode Unicode U+000A, umumnya disebut sebagai baris baru).

```

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 adalah algoritma yang didefinisikan oleh [RFC 2104 - Keyed-Hashing untuk Otentikasi Pesan](#). Algoritma tersebut menerima dua string byte sebagai input, yaitu kunci dan pesan. Untuk otentikasi permintaan Amazon S3, gunakan kunci akses AWS rahasia Anda (YourSecretAccessKey) sebagai kunci, dan peng UTF-8 kodean StringToSign sebagai pesan. Output dari juga HMAC-SHA1 merupakan string byte, yang disebut Digest. Parameter permintaan Signature dibangun berdasarkan Base64 yang mengkodekan digest ini.

Kanonikalisasi permintaan untuk penandatanganan

Ingatlah bahwa ketika sistem menerima permintaan yang diautentikasi, sistem tersebut membandingkan tanda tangan permintaan yang dikomputasi dengan tanda tangan yang disertakan dalam permintaan di StringToSign. Oleh karena itu, Anda harus melakukan komputasi tanda tangan dengan menggunakan metode yang sama seperti yang digunakan oleh Amazon S3. Proses menempatkan permintaan dalam bentuk yang disepakati untuk menandatangani kanonikalisasi.

Membangun CanonicalizedResource elemen

CanonicalizedResource mewakili sumber daya Amazon S3 yang ditarget oleh permintaan. Buat elemen permintaan REST sebagai berikut:

1. Mulai dengan string kosong ("").
2. Jika permintaan menetapkan bucket untuk menggunakan header Host HTTP (gaya hosting virtual), tambahkan nama bucket yang diawali dengan "/" (misalnya, "/bucketname"). Untuk

permintaan yang menggunakan gaya penulisan path dan permintaan yang tidak ditujukan ke bucket, jangan lakukan apa pun. Untuk informasi selengkapnya tentang permintaan gaya host virtual, lihat [Hosting virtual bucket](#).

Untuk permintaan gaya host virtual, "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg" adalah "/awsexamplebuck CanonicalizedResource et1".

Untuk permintaan gaya jalur, "https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg", CanonicalizedResource adalah "".

3. Tambahkan bagian jalur dari HTTP yang tidak didekode Request-URI, hingga tetapi tidak termasuk string kueri.

Untuk permintaan gaya host virtual, "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg" CanonicalizedResource adalah "/awsexamplebucket1/photos/puppy.jpg".

Untuk permintaan gaya jalur, "https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg", CanonicalizedResource adalah "/awsexamplebucket1/photos/puppy.jpg". Pada titik ini, CanonicalizedResource sama, baik untuk permintaan gaya penulisan hosting virtual dan permintaan gaya penulisan path.

Untuk permintaan yang tidak menyebutkan bucket, misalnya [Layanan DAPATKAN](#), tambahkan "/".

4. Jika permintaan membahas sub-sumber daya, seperti ?versioning, ?location, ?acl, ?lifecycle, atau ?versionid tambahkan sub-sumber dayanya, nilainya, jika ada, dan tanda tanya. Perhatikan bahwa dalam kasus beberapa subsumber daya, subsumber daya harus diurutkan secara leksikografis berdasarkan nama subsumber daya dan dipisahkan oleh '&', misalnya, ? ACL&versionId =value.

Subsumber daya yang harus disertakan saat membangun CanonicalizedResource Elemen adalah acl, siklus hidup, lokasi, logging, notifikasi, partNumber, kebijakan, requestPayment, uploadID, unggahan, versionId, versioning, versi, dan situs web.

Jika permintaan menentukan parameter string kueri yang menimpa nilai header respons (lihat [Dapatkan Objek](#)), tambahkan parameter string kueri dan nilainya. Saat menandatangani, Anda tidak melakukan encoding atas nilai-nilai ini; namun, saat membuat permintaan, Anda harus melakukan encoding atas nilai-nilai parameter ini. Parameter string kueri dalam permintaan DAPATKAN meliputi response-content-type, response-content-language, response-expires, response-cache-control, response-content-disposition, dan response-content-encoding.

Parameter string delete kueri harus disertakan saat Anda membuat permintaan Delete multi-objek. CanonicalizedResource

Elemen CanonicalizedResource yang berasal dari HTTP Request-URI harus ditandatangani secara harfiah seperti yang muncul dalam permintaan HTTP, termasuk karakter URL-Encoding meta.

Ini CanonicalizedResource mungkin berbeda dari HTTP Request-URI. Secara khusus, jika permintaan Anda menggunakan Host header HTTP untuk menentukan bucket, bucket tidak muncul di HTTP Request-URI. Namun, CanonicalizedResource harus tetap menyertakan bucket. Parameter string kueri mungkin juga muncul di Request-URI tetapi tidak termasuk dalam CanonicalizedResource. Untuk informasi selengkapnya, lihat Hosting [virtual bucket](#).

Membangun CanonicalizedAmzHeaders elemen

Untuk membangun CanonicalizedAmzHeaders bagian dariStringToSign, pilih semua header permintaan HTTP yang dimulai dengan 'x-amz-' (menggunakan perbandingan case-insensitive), dan gunakan proses berikut.

1. Ubah setiap nama header HTTP menjadi huruf kecil. Misalnya, 'X-Amz-Date' menjadi 'x-amz-date'.
2. Urutkan pengumpulan header secara leksikografis berdasarkan nama header.
3. Gabungkan kolom header dengan nama yang sama menjadi satu pasangan "header-name:comma-separated-value-list" sebagaimana yang ditentukan dalam RFC 2616, bagian 4.2, tanpa spasi antar nilai. Misalnya, dua metadata header 'x-amz-meta-username: fred' dan 'x-amz-meta-username: barney' akan digabungkan ke dalam header tunggal 'x-amz-meta-username: fred,barney'.
4. “Buka” header panjang yang mencakup beberapa baris (sebagaimana yang diizinkan oleh RFC 2616, bagian 4.2) dengan mengganti spasi terlipat (termasuk baris baru) dengan satu spasi.
5. Hapus setiap spasi di sekitar titik dua yang terdapat dalam header. Misalnya, header 'x-amz-meta-username: fred,barney' akan menjadi 'x-amz-meta-username:fred,barney'.
6. Terakhir, tambahkan karakter baris baru (U+000A) untuk setiap header kanonikalisasi dalam daftar hasil. Membuat elemen CanonicalizedResource dengan menggabungkan semua header dalam daftar ini menjadi satu string tunggal.

Elemen header StringToSign HTTP posisi versus bernama

Beberapa elemen header pertama dari `StringToSign` (`Content-Type`, `Date`, and `Content-MD5`) bersifat posisional. `StringToSign` tidak menyertakan nama-nama header ini, hanya nilainya dari permintaan. Sebaliknya, elemen `'x-amz-'` diberi nama. Baik nama header dan nilai header akan muncul di `StringToSign`.

Jika header posisional yang disebutkan dalam definisi `StringToSign` tidak ada dalam permintaan Anda (misalnya, `Content-Type` atau `Content-MD5` bersifat opsional untuk permintaan LETAKKAN dan tidak penting untuk permintaan DAPATKAN), ganti string kosong ("") untuk posisi tersebut.

Ketentuan stempel waktu

Stempel waktu yang valid (baik menggunakan header `HTTP Date` atau alternatif `x-amz-date`) wajib untuk permintaan yang diautentikasi. Selain itu, stempel waktu klien yang disertakan dengan permintaan yang diautentikasi harus dalam waktu 15 menit dari waktu sistem Amazon S3 saat permintaan tersebut diterima. Jika tidak, permintaan akan gagal dengan kode kesalahan `RequestTimeTooSkewed`. Tujuan dari pembatasan ini adalah untuk membatasi kemungkinan bahwa permintaan yang diintersepsi dapat diputar ulang oleh pihak lawan. Untuk perlindungan yang lebih kuat terhadap intervensi, gunakan transportasi HTTPS untuk permintaan yang diautentikasi.

Note

Batasan validasi pada tanggal permintaan hanya berlaku untuk permintaan yang diautentikasi yang tidak menggunakan autentikasi string kueri. Untuk informasi selengkapnya, lihat [Alternatif autentikasi permintaan string kueri](#).

Beberapa pustaka klien HTTP tidak memaparkan kemampuan untuk mengatur header `Date` untuk permintaan. Jika Anda mengalami masalah saat menyertakan nilai header `'Tanggal'` di header kanonikalisasi, Anda dapat mengatur stempel waktu untuk permintaan tersebut dengan menggunakan header `'x-amz-date'`. Nilai `x-amz-date` header harus dalam salah satu format RFC 2616 () <http://www.ietf.org/rfc/rfc2616.txt>. Saat header `x-amz-date` terdapat dalam permintaan, maka sistem akan mengabaikan header `Date` saat melakukan komputasi tanda tangan permintaan. Oleh karena itu, jika Anda menyertakan header `x-amz-date`, gunakan string kosong untuk `Date` saat membangun `StringToSign`. Lihat contohnya di bagian berikut.

Contoh-contoh autentikasi

Contoh-contoh dalam bagian ini menggunakan kredensial (nonkerja) dalam tabel berikut.

Parameter	Nilai
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecret AccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

Dalam contoh `StringToSign`, format tidak signifikan, dan `\n` berarti titik kode Unicode U+000A yang biasanya disebut sebagai baris baru. Selain itu, contoh tersebut menggunakan “+0000” untuk menentukan zona waktu. Anda dapat menggunakan “GMT” untuk menentukan zona waktu, tetapi tanda tangan yang ditunjukkan dalam contoh tersebut akan berbeda.

Objek DAPATKAN

Contoh ini mendapatkan objek dari bucket `awsexamplebucket1`.

Permintaan	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 AKIAIOSFODNN7EXAMPLE: qgk2+6Sv9/oM7G3qLEjTH1a1l1g=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:36:42 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

Perhatikan bahwa `CanonicalizedResource` menyertakan nama bucket, tetapi HTTP Request-URI tidak. (Bucket ditentukan berdasarkan header `Host`.)

Note

Skrip Python berikut ini menghitung tanda tangan sebelumnya, menggunakan parameter yang disediakan. Anda dapat menggunakan skrip ini untuk membuat tanda tangan Anda sendiri, mengganti kunci dan StringToSign sebagaimana mestinya.

```
import base64
import hmac
from hashlib import sha1

access_key = 'AKIAIOSFODNN7EXAMPLE'.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()}")
```

Objek LETAKKAN

Contoh ini menempatkan sebuah objek ke dalam bucket awsexamplebucket1.

Permintaan	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:</pre>	<pre>PUT\n \n image/jpeg\n Tue, 27 Mar 2007 21:15:45 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

Permintaan	StringToSign
<i>iqRzw+ileNPu1fhspnRs8n0jjIA=</i>	

Perhatikan Content-Type header dalam permintaan dan di StringToSign. Perhatikan juga bahwa Content-MD5 dibiarkan kosong di StringToSign, karena tidak ada dalam permintaan.

Daftar

Contoh ini mencantumkan konten dari bucket `awsexamplebucket1`.

Permintaan	StringToSign
<pre>GET /?prefix=photos&max-keys=50&marker=puppy HTTP/1.1 User-Agent: Mozilla/5.0 Host: awsexamplebucket1.s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 19:42:41 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: m0WP8eCtspQl5Ahe6L1SozdX9YA=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:42:41 +0000\n /awsexamplebucket1/</pre>

Perhatikan garis miring pada CanonicalizedResource dan tidak adanya parameter string kueri.

Mengambil

Contoh ini mengambil sub-sumber daya kebijakan kontrol akses untuk bucket `'awsexamplebucket1'`.

Permintaan	StringToSign
<pre>GET /?acl HTTP/1.1 Host: awsexamplebucket1.s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 19:44:46 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE:</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:44:46 +0000\n</pre>

Permintaan	StringToSign
<i>82ZHiFIjc+WbcwFKGUVEQspPn+0=</i>	/awsexamplebucket1/?acl

Perhatikan bagaimana parameter string kueri sub-sumber daya disertakan di CanonicalizedResource.

Delete

Contoh ini menghapus objek dari bucket 'awsexamplebucket1' menggunakan alternatif gaya penulisan path dan Tanggal.

Permintaan	StringToSign
<pre>DELETE /awsexamplebucket1/photos/puppy.jpg HTTP/1.1 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>DELETE\n \n \n Tue, 27 Mar 2007 21:20:26 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

Perhatikan bagaimana metode alternatif 'x-amz-date' untuk menentukan tanggal (karena perpustakaan klien kami mencegah kami mengatur tanggal, katakanlah). Dalam hal ini, x-amz-date lebih diutamakan dari header Date. Oleh karena itu, entri tanggal pada tanda tangan harus berisi nilai header x-amz-date.

Unggah

Contoh ini mengunggah objek ke bucket yang dihosting virtual bergaya CNAME dengan metadata.

Permintaan	StringToSign
PUT /db-backup.dat.gz HTTP/1.1	PUT\n

Permintaan	StringToSign
<pre>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 Authorization: AWS AKIAIOSFODNN7EXAMPLE: jtBQa0Aq+DkULFI8qrpwIjGEx0E=</pre>	<pre>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>

Perhatikan bagaimana header 'x-amz-' diurutkan, dipotong spasi ekstranya, dan diubah menjadi huruf kecil. Perhatikan juga bahwa beberapa header yang memiliki nama yang sama telah digabungkan menggunakan koma untuk memisahkan nilai.

Perhatikan bagaimana hanya header entitas HTTP Content-Type dan Content-MD5 yang muncul di StringToSign. Header entitas Content-* lainnya tidak muncul.

Sekali lagi, perhatikan CanonicalizedResource bahwa menyertakan nama bucket, tetapi HTTP Request-URI tidak. (Bucket ditentukan berdasarkan header Host.)

Tuliskan semua bucket saya

Permintaan	StringToSign
<pre>GET / HTTP/1.1 Host: s3.us-west-1.amazonaws.com Date: Wed, 28 Mar 2007 01:29:59 +0000</pre>	<pre>GET\n \n \n</pre>

Permintaan	StringToSign
<i>Authorization: AWS AKIAIOSFODNN7EXAMPLE:qGdzdERIC03wnaRNKh60qZehG9s=</i>	Wed, 28 Mar 2007 01:29:59 +0000\n /

Kunci Unicode

Permintaan	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:DNEZGsoieTZ92F3bUfSPQcbGmLM=</i>	GET\n \n \n Wed, 28 Mar 2007 01:49:49 +0000\n /dictionary/fran%C3%A7ais/pr %c3%a9f%c3%a8re

Note

Unsur-unsur StringToSign yang berasal dari diambil Request-URI secara harfiah, termasuk URL-Encoding dan kapitalisasi.

Masalah penandatanganan permintaan REST

Saat autentikasi permintaan REST gagal, sistem akan merespons permintaan tersebut dengan dokumen kesalahan XML. Informasi yang terdapat dalam dokumen kesalahan ini ditujukan untuk membantu pengembang melakukan diagnosis masalah. Secara khusus, elemen StringToSign dari dokumen kesalahan SignatureDoesNotMatch memberi tahu Anda dengan tepat kanonikalisasi permintaan yang digunakan oleh sistem.

Beberapa toolkit diam-diam memasukkan header yang tidak Anda ketahui sebelumnya, misalnya menambahkan header Content-Type selama LETAKKAN. Dalam sebagian besar kasus ini, nilai header yang dimasukkan tetap konstan, sehingga Anda dapat menemukan header yang hilang dengan menggunakan alat seperti Ethereal atau tcpmon.

Alternatif autentikasi permintaan string kueri

Anda dapat melakukan autentikasi atas jenis permintaan tertentu dengan meneruskan informasi yang diperlukan sebagai parameter string kueri dan bukan dengan menggunakan header HTTP `Authorization`. Ini berguna untuk mengaktifkan akses peramban pihak ketiga langsung ke data Amazon S3 pribadi Anda tanpa mengajukan permintaan melalui proxy. Tujuannya adalah untuk membuat permintaan yang "sudah ditandatangani sebelumnya" dan mengkodekannya sebagai URL yang dapat diambil oleh peramban pengguna akhir. Selain itu, Anda dapat membatasi sebuah permintaan yang telah ditandatangani sebelumnya dengan menentukan waktu kedaluwarsa.

Untuk informasi selengkapnya tentang menggunakan parameter kueri untuk mengotentikasi permintaan, lihat Meng [autentikasi Permintaan: Menggunakan Parameter Kueri \(Versi AWS Tangan 4\)](#) di Referensi API Layanan Penyimpanan Sederhana Amazon. Untuk contoh penggunaan AWS SDK untuk menghasilkan URL yang ditandatangani sebelumnya, lihat [Berbagi objek dengan URL yang ditandatangani sebelumnya](#).

Membuat tanda tangan

Berikut ini adalah contoh permintaan REST Amazon S3 string kueri yang diautentikasi.

```
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
```

Metode autentikasi permintaan string kueri tidak memerlukan header HTTP khusus. Sebaliknya, elemen autentikasi yang diperlukan akan ditentukan sebagai parameter string kueri:

Nama parameter string kueri	Nilai contoh	Deskripsi
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE	ID kunci AWS akses Anda. Menentukan kunci akses AWS rahasia yang digunakan untuk menandatangani permintaan dan, secara tidak langsung, identitas

Nama parameter string kueri	Nilai contoh	Deskripsi
		pengembang yang membuat permintaan.
Expires	1141889120	Waktu berakhirnya tanda tangan, ditetapkan sebagai jumlah detik sejak epoch (00:00:00 UTC pada tanggal 1 Januari 1970). Permintaan yang diterima setelah waktu ini (berdasarkan waktu server) akan ditolak.
Signature	vjbyPxybdZaNmGa%2ByT272YEAiv4%3D	Pengkodean URL dari pengkodean Base64 HMAC-SHA1 dari StringToSign.

Metode autentikasi permintaan string kueri sedikit berbeda dari metode biasa tetapi hanya dalam bentuk format dari parameter permintaan Signature dan elemen StringToSign. Berikut ini adalah pseudo-grammar yang mengilustrasikan metode autentikasi permintaan string kueri.

```
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 adalah ID kunci akses AWS rahasia yang diberikan Amazon kepada Anda ketika Anda mendaftar untuk menjadi pengembang Amazon Web Service. Perhatikan bagaimana membuatnya cocok untuk penempatan dalam string kueri. Signature URL-Encoded Perhatikan juga bahwa dalam StringToSign, elemen posisi Date HTTP telah diganti dengan Expires. CanonicalizedAmzHeaders dan CanonicalizedResource sama.

Note

Dalam metode autentikasi string kueri, Anda tidak menggunakan Date atau x-amz-date request saat menghitung string yang akan ditandatangani.

Autentikasi permintaan string kueri

Permintaan	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>

Contoh ini mengasumsikan bahwa ketika browser membuat permintaan GET, itu tidak akan memberikan header Content-MD5 atau Content-Type header, juga tidak akan menyetel header x-amz-, sehingga bagian-bagian tersebut StringToSign dibiarkan kosong.

Menggunakan pengodean Base64

Tanda tangan permintaan HMAC harus dikodekan dengan Base64. Pengodean Base64 mengubah tanda tangan menjadi string ASCII sederhana yang dapat dilampirkan ke permintaan. Karakter yang dapat muncul dalam string tanda tangan seperti tambah (+), garis miring (/), dan sama dengan (=) harus dikodekan jika digunakan dalam URI. Misalnya, jika kode autentikasi menyertakan tanda tambah (+), konversikan kode tersebut sebagai %2B dalam permintaan. Konversikan kode garis miring menjadi %2F dan sama dengan menjadi %3D.

Untuk contoh pengodean Base64, lihat Amazon S3 [Contoh-contoh autentikasi](#).

Browser-based unggah menggunakan POST (AWS tanda tangan versi 2)

Amazon S3 mendukung POST, yang memungkinkan pengguna mengunggah konten secara langsung ke Amazon S3. POST dirancang untuk menyederhanakan unggahan, mengurangi latensi

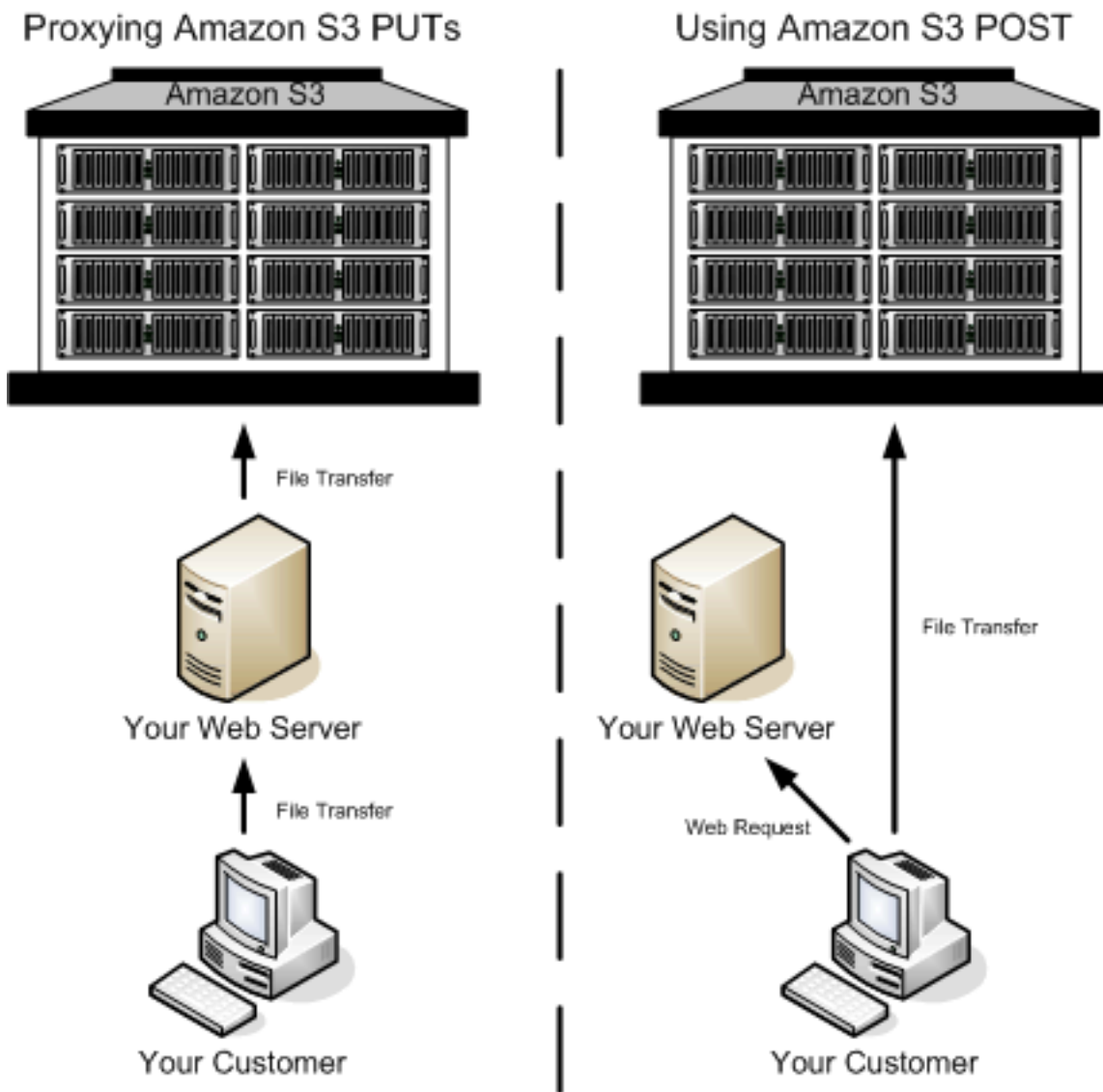
unggahan, dan menghemat biaya Anda pada aplikasi tempat pengguna mengunggah data untuk disimpan di Amazon S3.

 Note

Otentikasi permintaan yang dibahas di bagian ini didasarkan pada AWS Signature Version 2, sebuah protokol untuk mengotentikasi permintaan API masuk ke AWS layanan.

Amazon S3 sekarang mendukung Signature Version 4, protokol untuk mengotentikasi permintaan API masuk ke AWS layanan, secara keseluruhan. Wilayah AWS Pada saat ini, Wilayah AWS dibuat sebelum 30 Januari 2014 akan terus mendukung protokol sebelumnya, Signature Version 2. Wilayah baru setelah tanggal 30 Januari 2014 hanya akan mendukung Tanda Tangan Versi 4 dan oleh karena itu semua permintaan ke wilayah tersebut harus dilakukan dengan Tanda Tangan Versi 4. Untuk informasi selengkapnya, lihat [Mengotentikasi Permintaan di Ung Browser-Based gahan Menggunakan POST \(Tanda T AWS angan Versi 4\)](#) di Referensi API Layanan Penyimpanan Sederhana Amazon.

Gambar berikut menunjukkan unggahan menggunakan Amazon S3 POST.



Mengunggah menggunakan POST

- 1 Pengguna membuka peramban web dan mengakses halaman web Anda.
- 2 Halaman web Anda berisi formulir HTTP yang memuat semua informasi yang diperlukan pengguna untuk mengunggah konten ke Amazon S3.
- 3 Pengguna mengunggah konten secara langsung ke Amazon S3.

Note

Autentikasi string kueri tidak didukung untuk POST.

Formulir HTML (AWS tanda tangan versi 2)

Topik

- [Pengodean formulir HTML](#)
- [Deklarasi formulir HTML](#)
- [Bidang formulir HTML](#)
- [Konstruksi kebijakan](#)
- [Membuat tanda tangan](#)
- [Pengalihan](#)

Saat berkomunikasi dengan Amazon S3, Anda biasanya menggunakan REST atau SOAP API untuk menjalankan, mendapatkan, menghapus, dan melakukan operasi lainnya. Dengan POST, pengguna mengunggah data secara langsung ke Amazon S3 melalui peramban mereka, yang tidak dapat memproses SOAP API atau membuat permintaan REST PUT.

Note

SOAP API untuk Amazon S3 tidak tersedia untuk pelanggan baru, dan mendekati End of Life (EOL) pada 31 Agustus 2025. Kami menyarankan Anda menggunakan REST API atau AWS SDK.

Agar pengguna dapat mengunggah konten ke Amazon S3 dengan menggunakan peramban mereka, Anda perlu menggunakan formulir HTML. Formulir HTML terdiri dari deklarasi formulir dan bidang formulir. Deklarasi formulir berisi informasi tingkat tinggi mengenai permintaan tersebut. Kolom formulir berisi informasi terperinci tentang permintaan, serta kebijakan yang digunakan untuk mengautentikasi permintaan tersebut dan memastikan bahwa permintaan tersebut memenuhi persyaratan yang Anda tentukan.

Note

Data formulir dan batasannya (tidak termasuk isi file) tidak boleh lebih dari 20 KB.

Bagian ini menjelaskan cara menggunakan formulir HTML.

Pengodean formulir HTML

Bentuk dan kebijakan harus di UTF-8 kodekan. Anda dapat menerapkan UTF-8 pengkodean ke formulir dengan menentukannya di judul HTML atau sebagai header permintaan.

Note

Deklarasi formulir HTML tidak menerima parameter-parameter autentikasi string kueri.

Berikut ini adalah contoh peng UTF-8 kodean dalam judul HTML:

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

Berikut ini adalah contoh peng UTF-8 kodean dalam header permintaan:

```
Content-Type: text/html; charset=UTF-8
```

Deklarasi formulir HTML

Deklarasi formulir memiliki tiga komponen, yakni: aksi, metode, dan jenis enklosure. Jika salah satu dari nilai ini tidak sesuai, maka permintaan tersebut gagal.

Tindakan ini menentukan URL yang memproses permintaan, yang harus diatur ke URL bucket. Sebagai contoh, jika nama bucket Anda adalah `awsexamplebucket1` dan Wilayahnya adalah AS Barat (California Utara), URL-nya adalah `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/`.

Note

Nama kunci ditentukan dalam bidang formulir.

Metodenya harus POST.

Jenis penutup (enctype) harus ditentukan dan harus disetel ke multipart/form-data untuk unggahan file dan unggahan area teks. Untuk informasi selengkapnya, buka [RFC 1867](#).

Example

Contoh berikut adalah deklarasi formulir untuk bucket "awsexamplebucket1".

```
<form action="https://awsexamplebucket1.s3.us-west-1.amazonaws.com/" method="post"
enctype="multipart/form-data">
```

Bidang formulir HTML

Tabel berikut menjelaskan bidang yang dapat digunakan dalam sebuah formulir HTML.

Note

Variabel `${filename}` secara otomatis diganti dengan nama file yang diberikan oleh pengguna dan dikenali oleh semua bidang formulir. Jika peramban atau klien menyediakan jalur penuh atau sebagian ke file, maka hanya teks yang ada setelah garis miring terakhir (/) atau garis miring terbalik (\) yang akan digunakan. Sebagai contoh, "C:\Program Files\directory1\file.txt" akan diinterpretasikan sebagai "file.txt". Jika tidak ada nama file atau file yang diberikan, maka variabel diganti dengan string kosong.

Nama bidang	Deskripsi	Diperlukan
AWSAccessKeyId	ID K AWS unci Akses pemilik bucket yang memberikan akses pengguna anonim untuk permintaan yang memenuhi kumpulan batasan dalam kebijakan. Kolom ini wajib diisi jika permintaan menyertakan dokumen kebijakan.	Bersyarat
acl	Daftar kontrol akses (ACL) Amazon S3. Jika ternyata daftar kontrol akses yang ditentukan	Tidak

Nama bidang	Deskripsi	Diperlukan
	n tidak valid, maka akan muncul pesan kesalahan. Tipe: String Default: privat Nilai Valid: 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 tajuk. Untuk informasi selengkapnya, lihat PUT Objek .	Tidak
key	Nama kunci yang diunggah. Untuk menggunakan nama file yang diberikan oleh pengguna, gunakan variabel \${filename}. Misalnya, jika pengguna Betty mengunggah file lolcatz.jpg dan Anda menentukan/user/betty/\${filename}, file tersebut disimpan sebagai//lolcatz.jpg. user/betty Untuk informasi lebih lanjut, lihat Bekerja dengan objek metadata.	Ya
policy	Kebijakan keamanan yang menjelaskan apa yang diizinkan dalam permintaan. Permintaan yang tidak memiliki kebijakan keamanan dianggap anonim dan hanya akan berhasil pada bucket yang dapat ditulis oleh publik.	Tidak

Nama bidang	Deskripsi	Diperlukan
success_action_redirect, redirect	URL yang akan diarahkan klien setelah unggahan berhasil. Amazon S3 menambahkan nilai bucket, kunci, dan etag sebagai parameter string kueri ke URL. Jika success_action_redirect tidak ditentukan, maka Amazon S3 akan mengembalikan jenis dokumen kosong yang ditentukan dalam bidang success_action_status. Jika Amazon S3 tidak dapat menginterpretasikan URL, maka Amazon S3 akan mengabaikan bidang tersebut. Jika pengunggahan gagal, maka Amazon S3 akan menampilkan pesan kesalahan dan tidak akan mengalihkan pengguna ke URL. Untuk informasi selengkapnya, lihat Pengalihan.  Note Nama bidang pengalihan telah dihilangkan dan dukungan untuk nama bidang pengalihan akan dihapus di masa mendatang.	Tidak

Nama bidang	Deskripsi	Diperlukan
success_action_status	Kode status dikembalikan kepada klien setelah berhasil mengunggah jika success_action_redirect tidak ditentukan. Nilai yang valid adalah 200, 201, atau 204 (default). Jika nilai diatur ke 200 atau 204, maka Amazon S3 akan mengembalikan dokumen kosong dengan kode status 200 atau 204. Jika nilai diatur ke 201, maka Amazon S3 akan mengembalikan dokumen XML dengan kode status 201. Untuk informasi tentang konten dokumen XML, lihat Objek POST. Jika nilai tidak diatur atau jika diatur dengan nilai yang tidak valid, maka Amazon S3 akan mengembalikan dokumen kosong dengan kode status 204.	Tidak

 **Note**

Beberapa versi dari Adobe Flash player tidak menangani respons HTTP dengan baik pada bodi kosong. Untuk mendukung pengunggahan melalui Adobe Flash, sebaiknya atur success_action_status menjadi 201.

Nama bidang	Deskripsi	Diperlukan
signature	Tanda tangan HMAC dibangun dengan menggunakan kunci akses rahasia yang sesuai dengan yang disediakan AWS Access Key Id. Bidang ini wajib diisi jika dokumen kebijakan disertakan bersama permintaan. Untuk informasi selengkapnya, lihat Manajemen Identitas dan Akses untuk Amazon S3.	Bersyarat
x-amz-security-token	Token keamanan digunakan oleh kredensial sesi Jika permintaan menggunakan Amazon DevPay maka diperlukan dua bidang x-amz-security-token formulir: satu untuk token produk dan satu untuk token pengguna. Jika permintaan menggunakan kredensial sesi, maka permintaan tersebut memerlukan satu formulir x-amz-security-token . Untuk informasi selengkapnya, lihat Kredensial Keamanan Sementara dalam Panduan Pengguna IAM.	Tidak
Nama bidang lain yang diawali dengan x-amz-meta	User-specified metadata. Amazon S3 tidak memvalidasi atau menggunakan data ini. Untuk informasi selengkapnya, lihat PUT Objek.	Tidak

Nama bidang	Deskripsi	Diperlukan
file	Konten file atau teks. File atau konten harus menjadi bidang terakhir dalam formulir. Bidang apa pun di bawahnya akan diabaikan. Anda tidak dapat mengunggah lebih dari satu file sekaligus.	Ya

Konstruksi kebijakan

Topik

- [Kedaluwarsa](#)
- [Ketentuan](#)
- [Pencocokan ketentuan](#)
- [Pelarian karakter](#)

Kebijakan adalah dokumen UTF-8 dan Base64-encoded JSON yang menentukan kondisi yang harus dipenuhi permintaan dan digunakan untuk mengotentikasi konten. Tergantung pada cara Anda merancang dokumen kebijakan, Anda dapat menggunakannya untuk setiap unggahan, setiap pengguna, untuk semua unggahan, atau sesuai dengan desain lain yang memenuhi kebutuhan Anda.

Note

Meskipun dokumen kebijakan bersifat opsional, seabiknya buat bucket yang dapat ditulis secara publik.

Berikut ini adalah contoh dari dokumen kebijakan:

```
{ "expiration": "2007-12-01T12:00:00.000Z",  
  
  "conditions": [
```

```
{
  "acl": "public-read" },
  "bucket": "awsexamplebucket1" },
  ["starts-with", "$key", "user/eric/"],
]
}
```

Dokumen kebijakan berisi masa kedaluwarsa dan ketentuan-ketentuan.

Kedaluwarsa

Elemen kedaluwarsa menetapkan tanggal kedaluwarsa kebijakan dengan format tanggal ISO 8601 UTC. Misalnya, “2007-12-01T12:00:00.000Z” menentukan bahwa kebijakan tersebut tidak valid setelah tengah malam UTC pada 2007-12-01. Kedaluwarsa wajib ada dalam kebijakan.

Ketentuan

Ketentuan dalam dokumen kebijakan memvalidasi konten dari objek yang diunggah. Setiap bidang formulir yang Anda tentukan dalam formulir (kecuali `AWSAccessKeyId`, tanda tangan, file, kebijakan, dan nama bidang yang memiliki awalan `x-ignore-`) harus disertakan dalam daftar kondisi.

Note

Jika Anda memiliki beberapa bidang dengan nama yang sama, maka nilai-nilainya harus dipisahkan dengan koma. Contoh, jika Anda memiliki dua bidang bernama “x-amz-meta-tag” dan yang pertama memiliki nilai “Ninja” dan yang kedua memiliki nilai “Stallman”, maka Anda harus mengatur dokumen kebijakan menjadi `Ninja, Stallman`.

Semua variabel yang ada dalam formulir diperluas sebelum kebijakan tersebut divalidasi. Oleh karena itu, semua pencocokan kondisi harus dilakukan pada kolom-kolom yang diperluas. Misalnya, jika Anda mengatur bidang kunci ke `user/betty/${filename}`, kebijakan Anda mungkin `[ "starts-with", "$key", "user/betty/" ]`. Jangan masukkan `[ "starts-with", "$key", "user/betty/${filename}" ]`. Untuk informasi selengkapnya, lihat [Pencocokan ketentuan](#).

Tabel berikut menjelaskan ketentuan dokumen kebijakan.

Nama elemen	Deskripsi
acl	Menentukan ketentuan yang harus dipenuhi oleh ACL. Dukung pencocokan yang tepat dan <code>starts-with</code> .
content-length-range	Menentukan ukuran minimal dan maksimal yang diizinkan untuk konten yang diunggah. Dukung penyesuaian rentang.
Cache-Control,, Content-Type, Content-DispositionContent-Encoding, Kedaluwarsa	REST-specific tajuk. Dukung pencocokan yang tepat dan <code>starts-with</code> .
kunci	Nama kunci yang diunggah. Dukung pencocokan yang tepat dan <code>starts-with</code> .
success_action_redirect, mengalihkan	URL yang akan diarahkan klien setelah unggahan berhasil. Dukung pencocokan yang tepat dan <code>starts-with</code> .
success_action_status	Kode status dikembalikan kepada klien setelah berhasil mengunggah jika <code>success_action_redirect</code> tidak ditentukan. Dukung pencocokan yang tepat.
x-amz-security-token	Token DevPay keamanan Amazon. Setiap permintaan yang menggunakan Amazon DevPay memerlukan dua bidang <code>x-amz-security-token</code> formulir: satu untuk token produk dan satu untuk token pengguna. Akibatnya, nilai harus dipisahkan dengan koma. Contoh, jika token pengguna adalah <code>eW91dHVhZQ==</code> dan token produk adalah <code>b0hnNVNKWVJIQTA=</code> , maka

Nama elemen	Deskripsi
	Anda harus mengatur entri kebijakan menjadi: { "x-amz-security-token": "ew91dHVhZQ==,b0hnNVNKWVJIQTA=" } .
Nama bidang lain yang diawali dengan x-amz-meta	User-specified metadata. Dukung pencocokan yang tepat dan starts-with .

Note

Jika alat memberikan bidang tambahan (misalnya, Flash menambahkan nama file), maka Anda harus menambahkannya pada dokumen kebijakan. Jika Anda dapat mengontrol fungsi ini, tambahkan awalan `x-ignore-` pada bidang tersebut sehingga Amazon S3 mengabaikan fitur tersebut dan tidak akan memengaruhi versi fitur ini di masa mendatang.

Pencocokan ketentuan

Tabel berikut menjelaskan jenis pencocokan ketentuan. Meskipun Anda harus menentukan satu ketentuan untuk setiap bidang formulir yang disebutkan dalam formulir tersebut, Anda dapat membuat kriteria pencocokan yang lebih kompleks dengan menetapkan beberapa ketentuan untuk bidang formulir.

Ketentuan	Deskripsi
Kecocokan yang Tepat	Kecocokan yang tepat memverifikasi bahwa kolom cocok dengan nilai spesifik. Contoh ini menunjukkan bahwa ACL harus diatur ke baca publik: <pre>{"acl": "public-read" }</pre> Contoh ini adalah cara alternatif untuk menunjukkan bahwa ACL harus diatur ke baca publik:

Ketentuan	Deskripsi
	<pre>["eq", "\$acl", "public-read"]</pre>
Dimulai Dengan	Jika nilai harus dimulai dengan nilai tertentu, gunakan mulai-dengan. Contoh ini menunjukkan bahwa kunci harus dimulai dengan user/betty: <pre>["starts-with", "\$key", "user/betty/"]</pre>
Mencocokkan Konten Apa Pun	Untuk mengonfigurasi kebijakan yang mengizinkan konten apa pun di dalam sebuah bidang, gunakan mulai-dengan dengan nilai kosong. Contoh ini memungkinkan success_action_redirect: <pre>["starts-with", "\$success_action_redirect", ""]</pre>
Menentukan Rentang	Untuk kolom yang menerima rentang, pisahkan rentang atas dan bawah dengan koma. Contoh ini memungkinkan ukuran file dari 1 hingga 10 megabyte: <pre>["content-length-range", 1048579, 10485760]</pre>

Pelarian karakter

Tabel berikut menjelaskan pelarian karakter dari dalam dokumen kebijakan.

Karakter pelarian	Deskripsi
\\	Garis miring terbalik
\\\$	Tanda dolar

Karakter pelarian	Deskripsi
<code>\b</code>	Spasi Mundur
<code>\f</code>	Umpan formulir
<code>\n</code>	Baris baru
<code>\r</code>	Kembali ke awal baris
<code>\t</code>	Tab Horizontal
<code>\v</code>	Tab vertikal
<code>\ u <i>xxxx</i></code>	Semua karakter Unicode

Membuat tanda tangan

Langkah	Deskripsi
1	Enkodekan kebijakan dengan menggunakan UTF-8.
2	Enkode UTF-8 byte tersebut dengan menggunakan Base64.
3	Tanda tangani kebijakan dengan kunci akses rahasia Anda dengan menggunakan HMACSHA-1.
4	Enkode SHA-1 tanda tangan dengan menggunakan Base64.

Untuk informasi umum tentang otentikasi, lihat Manajemen [Identitas dan Akses untuk Amazon S3](#).

Pengalihan

Bagian ini menjelaskan cara menangani pengalihan.

Pengalihan umum

Setelah menyelesaikan permintaan POST, pengguna akan dialihkan ke lokasi yang telah Anda tentukan dalam bidang `success_action_redirect`. Jika Amazon S3 tidak dapat menginterpretasikan URL, maka Amazon S3 mengabaikan bidang `success_action_redirect`.

Jika `success_action_redirect` tidak ditentukan, maka Amazon S3 akan mengembalikan jenis dokumen kosong yang ditentukan dalam bidang `success_action_status`.

Jika permintaan POST gagal, maka Amazon S3 akan menampilkan kesalahan dan tidak akan memberikan pengalihan.

Pre-upload pengalihan

Jika ember Anda dibuat menggunakan `<CreateBucketConfiguration>`, maka pengguna akhir Anda mungkin memerlukan pengalihan. Jika hal ini terjadi, beberapa peramban mungkin menangani pengalihan dengan tidak benar. Hal ini tergolong jarang terjadi, namun kemungkinan besar akan terjadi setelah pembuatan bucket.

Unggah contoh (AWS tanda tangan versi 2)

Topik

- [Unggahan file](#)
- [Unggahan bidang teks](#)

Note

Otentikasi permintaan yang dibahas di bagian ini didasarkan pada AWS Signature Version 2, sebuah protokol untuk mengotentikasi permintaan API masuk ke AWS layanan.

Amazon S3 sekarang mendukung Signature Version 4, protokol untuk mengotentikasi permintaan API masuk ke AWS layanan, secara keseluruhan. Wilayah AWS Pada saat ini, Wilayah AWS dibuat sebelum 30 Januari 2014 akan terus mendukung protokol sebelumnya, Signature Version 2. Wilayah baru setelah tanggal 30 Januari 2014 hanya akan mendukung Tanda Tangan Versi 4 dan oleh karena itu semua permintaan ke wilayah tersebut harus dilakukan dengan Tanda Tangan Versi 4. Untuk informasi selengkapnya, lihat [Contoh: Meng](#)

[Browser-Based unggah menggunakan HTTP POST \(Menggunakan T AWS anda Tangan Versi 4\)](#) di Referensi API Layanan Penyimpanan Sederhana Amazon.

Unggahan file

Contoh ini menunjukkan proses lengkap untuk membuat kebijakan dan formulir yang dapat digunakan untuk mengunggah lampiran file.

Pembuatan kebijakan dan formulir

Kebijakan berikut mendukung unggahan ke Amazon S3 untuk 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", ""]
  ]
}
```

Kebijakan ini mengharuskan:

- Unggahan harus dilakukan sebelum pukul 12:00 UTC tanggal 1 Desember 2007.
- Konten harus diunggah ke bucket `awsexamplebucket1`.
- Kuncinya harus dimulai dengan `"user/eric/"`.
- ACL diatur menjadi baca publik.
- `Success_action_redirect` disetel ke `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html`.
- Objek adalah file gambar.
- Tanda `x-amz-meta-uid` harus diatur menjadi `14365123651274`.
- Tanda `x-amz-meta` dapat berisi nilai apa pun.

Berikut ini adalah Base64-encoded versi kebijakan ini.

eyJ0eXhwaXJhdGlvbiI6IClYMDA3LTExVDEyOjAwMjAwLjAwMFoiLAogICJjb25kaXRpb25zIjoqWwoqICAgeyJidWN

Menggunakan kredensial Anda untuk membuat tanda tangan, misalnya

0RavWzkyqo6QX9caELEqKi9kDbU= adalah tanda tangan untuk dokumen kebijakan sebelumnya.

Formulir berikut mendukung permintaan POST ke amzn-s3-demo-bucket bucket yang menggunakan kebijakan ini.

```
<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>
```

Permintaan sampel

Permintaan ini mengasumsikan bahwa gambar yang diunggah adalah 117,108 byte; data gambar tidak disertakan.

```
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"

eyJhZXBwaXJhdGlvbiI6ICIyMDA3LTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWN
--9431149156168
Content-Disposition: form-data; name="Signature"

0RavWzkygo6QX9caELEqKi9kDbU=
--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--
```

Contoh respons

```
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
```

Unggahan bidang teks

Topik

- [Pembuatan kebijakan dan formulir](#)
- [Permintaan sampel](#)
- [Contoh respons](#)

Contoh berikut menunjukkan proses yang lengkap untuk membuat kebijakan dan formulir untuk mengunggah bidang teks. Mengunggah bidang teks berguna untuk mengirimkan konten yang dibuat pengguna, misalnya posting blog.

Pembuatan kebijakan dan formulir

Kebijakan berikut mendukung unggahan bidang teks ke Amazon S3 untuk 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", ""]
  ]
}
```

Kebijakan ini mengharuskan:

- Unggahan harus dilakukan sebelum pukul 12:00 GMT pada tanggal 1 Desember 2007.
- Konten harus diunggah ke bucket `awsexamplebucket1`.
- Kuncinya harus dimulai dengan `"user/eric/"`.
- ACL diatur menjadi baca publik.
- `Success_action_redirect` disetel ke `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html`.
- Objek adalah teks HTML.
- Tanda `x-amz-meta-uid` harus diatur menjadi `14365123651274`.
- Tanda `x-amz-meta` dapat berisi nilai apa pun.

Berikut ini adalah Base64-encoded versi kebijakan ini.

```
eyJhZiZlXhwaXJhdGlvbiI6IClYMDA3LlEYLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXR
```

```
pb25zIjogWwogICAgeyJidWNrZXQiOiAiam9obnNtaXR0In0sCiAgICBbInN0YXJ0cy13aXR0IiwgIiRrZXkiLCaidXNlci
LAogICAgeyJhY2wiOiAicHVibGljLXJlYWQifSwKICAgIHsic3VjY2Vzc19hY3Rpb25fcmVkaXJlY3QiOiAiaHR0cDovL2p
C5zMy5hbWw6b25hd3MuY29tL25ld19wb3N0Lmh0bWwifSwKICAgIFsiZXEiLCAiJENvbnRlbnQtVHlwZSI6ICJ0ZXh0L2h0
CAgIHsieC1hbXotbWV0YS11dWlkIjogIjE0MzY1MTIzNjUxMjc0In0sCiAgICBbInN0YXJ0cy13aXR0IiwgIiR4LWFtei1t
IsICIiXQogIF0KfQo=
```

Dengan menggunakan kredensial Anda, buat tanda tangan. Contohnya,

qA7FWXKq6VvU68lI9KdveT1cWgF= adalah tanda tangan untuk dokumen kebijakan sebelumnya.

Formulir berikut mendukung permintaan POST ke amzn-s3-demo-bucket bucket yang menggunakan kebijakan ini.

```
<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>
```


Permintaan sampel

Permintaan ini mengasumsikan bahwa gambar yang diunggah adalah 117,108 byte; data gambar tidak disertakan.

```
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"
eyJhZXBwaXJhdGlvbiI6ICIyMDA3LTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgZjIdWw
--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--

```

Contoh respons

```

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 dengan adobe flash (AWS tanda tangan versi 2)

Bagian ini menjelaskan cara menggunakan POST dengan Adobe Flash.

Keamanan Adobe flash player

Secara default, model keamanan Adobe Flash Player akan melarang Adobe Flash Player membuat koneksi jaringan ke server di luar domain yang melayani file SWF.

Untuk menimpa pengaturan default, Anda harus mengunggah file `crossdomain.xml` yang dapat dibaca publik ke bucket yang akan menerima unggahan POST. Berikut ini adalah sampel dari 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

Untuk informasi selengkapnya tentang model keamanan Adobe Flash, kunjungi situs web Adobe.

Menambahkan file `crossdomain.xml` ke bucket akan membuat Adobe Flash Player terhubung ke file `crossdomain.xml` dalam bucket Anda; namun, file ini tidak akan memberikan akses ke bucket Amazon S3 sebenarnya.

Pertimbangan Adobe flash

FileReference API di Adobe Flash menambahkan bidang `Filename` formulir ke permintaan POST. Saat membuat aplikasi Adobe Flash yang diunggah ke Amazon S3 menggunakan tindakan FileReference API, sertakan kondisi berikut dalam kebijakan Anda:

```
['starts-with', '$Filename', '']
```

Beberapa versi Adobe Flash Player tidak menangani dengan benar respons HTTP yang memiliki bodi kosong. Untuk mengonfigurasi POST guna mengembalikan respons yang tidak memiliki bodi kosong, atur `success_action_status` menjadi 201. Amazon S3 kemudian akan mengembalikan dokumen XML dengan kode status 201. Untuk informasi tentang konten dokumen XML, lihat [Objek POST](#). Untuk informasi tentang kolom formulir, lihat [Bidang formulir HTML](#).

Lampiran: API Konfigurasi Siklus Hidup (Tidak Digunakan)

Konfigurasi siklus hidup bucket diperbarui untuk mendukung filter berdasarkan tag objek. Artinya, Anda sekarang dapat menentukan aturan yang menentukan awalan nama kunci, satu atau lebih tag

objek, atau keduanya untuk memilih subset objek yang berlaku aturan. API telah diperbarui sesuai dengan itu. Topik berikut menjelaskan versi sebelumnya dari operasi siklus hidup bucket PUT dan GET untuk kompatibilitas mundur.

Topik

- [Siklus hidup PUT Bucket \(Tidak digunakan lagi\)](#)
- [Siklus hidup GET Bucket \(Tidak digunakan lagi\)](#)

Siklus hidup PUT Bucket (Tidak digunakan lagi)

Deskripsi

Important

Untuk versi terbaru dari API ini, lihat [PutBucketLifecycleConfiguration](#). Versi ini sudah tidak digunakan lagi. Konfigurasi siklus hidup yang ada akan berfungsi. Untuk konfigurasi siklus hidup baru, gunakan API yang diperbarui.

Membuat konfigurasi siklus hidup baru untuk bucket atau mengganti konfigurasi siklus hidup yang ada. Untuk informasi tentang konfigurasi siklus hidup, lihat [Manajemen Siklus Hidup Objek](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Izin

Secara default, semua sumber daya Amazon S3, termasuk bucket, objek, dan subsumber daya terkait (misalnya, konfigurasi siklus hidup dan konfigurasi situs web) bersifat pribadi. Hanya pemilik sumber daya, Akun AWS yang membuat sumber daya, yang dapat mengaksesnya. Pemilik sumber daya dapat secara opsional memberikan izin akses kepada orang lain dengan menulis kebijakan akses. Untuk operasi ini, pengguna harus mendapatkan `s3:PutLifecycleConfiguration` izin.

Anda juga dapat secara eksplisit menolak izin. Penolakan eksplisit juga menggantikan izin lainnya. Jika Anda ingin mencegah pengguna atau akun menghapus atau menghapus objek dari bucket Anda, Anda harus menolak izin mereka untuk tindakan berikut:

- `s3:DeleteObject`
- `s3:DeleteObjectVersion`
- `s3:PutLifecycleConfiguration`

Untuk informasi selengkapnya tentang izin, lihat [Mengelola Izin Akses ke Sumber Daya Amazon S3 Anda](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Permintaan

Sintaksis

```
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

Untuk detail tentang string otorisasi, lihat [Permintaan Otentikasi \(AWS Versi Tanda Tangan 4\)](#).

Parameter Permintaan

Implementasi operasi ini tidak menggunakan parameter permintaan.

Header Permintaan

Nama	Deskripsi	Diperlukan
Content-MD5	Intisari data MD5 128-bit yang dikodekan base64. Anda harus menggunakan header ini sebagai pemeriksaan integritas pesan untuk memverifikasi bahwa badan permintaan tidak rusak saat transit. Untuk informasi lebih lanjut, lihat RFC 1864. Tipe: String Default: Tidak Ada	Ya

Isi Permintaan

Dalam permintaan, Anda menentukan konfigurasi siklus hidup di badan permintaan. Konfigurasi siklus hidup ditentukan sebagai XML. Berikut ini adalah contoh konfigurasi siklus hidup dasar. Ini menentukan satu aturan. Aturan Prefix dalam mengidentifikasi objek yang berlaku aturan. Aturan ini juga menentukan dua tindakan (Transition dan Expiration). Setiap tindakan menentukan

garis waktu kapan Amazon S3 harus melakukan tindakan. Men Status unjukkan apakah aturan diaktifkan atau dinonaktifkan.

```
<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>
```

Jika status bucket Anda diaktifkan versi atau ditangguhkan versi, Anda dapat memiliki banyak versi dari objek yang sama: satu versi saat ini dan nol atau lebih versi noncurrent. Konfigurasi siklus hidup berikut menentukan tindakan (NoncurrentVersionTransition, NoncurrentVersionExpiration) yang khusus untuk versi objek noncurrent.

```
<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>
```

Anda dapat menggunakan unggahan multibagian API untuk mengunggah objek besar dalam beberapa bagian. Untuk informasi selengkapnya tentang unggahan multipart, lihat Ikhtisar Unggahan

[Multipart](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Dengan konfigurasi siklus hidup, Anda dapat memberi tahu Amazon S3 untuk membatalkan upload multipart yang tidak lengkap, yang diidentifikasi oleh awalan nama kunci yang ditentukan dalam aturan, jika tidak selesai dalam jumlah hari yang ditentukan. Saat Amazon S3 membatalkan unggahan multipart, Amazon S3 akan menghapus semua bagian yang terkait dengan unggahan. Ini memastikan bahwa Anda tidak memiliki unggahan multipart yang tidak lengkap yang meninggalkan bagian yang disimpan di Amazon S3, jadi Anda tidak perlu membayar biaya penyimpanan untuk itu. Berikut ini adalah contoh konfigurasi siklus hidup yang menentukan aturan dengan tindakan `AbortIncompleteMultipartUpload`. Tindakan ini memberitahu Amazon S3 untuk membatalkan unggahan multipart yang tidak lengkap tujuh hari setelah inisiasi.

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>SomeKeyPrefix</Prefix>
    <Status>rule-status</Status>
    <AbortIncompleteMultipartUpload>
      <DaysAfterInitiation>7</DaysAfterInitiation>
    </AbortIncompleteMultipartUpload>
  </Rule>
</LifecycleConfiguration>
```

Tabel berikut menjelaskan elemen XML dalam konfigurasi siklus hidup.

Nama	Deskripsi	Diperlukan
<code>AbortIncompleteMultipartUpload</code>	Wadah untuk menentukan kapan unggahan multipart yang tidak lengkap memenuhi syarat untuk operasi pembatalan. Anak: <code>DaysAfterInitiation</code> Tipe: Kontainer Leluhur: <code>Rule</code>	Ya, jika tidak ada tindakan lain yang ditentukan untuk aturan
Date	Tanggal kapan Anda ingin Amazon S3 mengambil tindakan. Untuk informasi	Ya, jika Days

Nama	Deskripsi	Diperlukan
	selengkapnya, lihat Aturan Siklus Hidup: Berdasarkan Tanggal Ter tentu di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Nilai tanggal harus sesuai dengan format ISO 8601. Waktunya adalah selalu tengah malam UTC. Tipe: String Leluhur: <code>Expiration</code> atau <code>Transition</code>	dan tidak <code>ExpiredObjectDeleteMarker</code> ada
Days	Menentukan jumlah hari setelah pembuatan objek ketika tindakan aturan tertentu berlaku. Jenis: Integer Nonnegatif saat digunakan dengan <code>Transition</code> , Integer Positif saat digunakan dengan <code>Expiration</code> Leluhur: <code>Expiration</code> , <code>Transition</code>	Ya, jika <code>Date</code> dan tidak <code>ExpiredObjectDeleteMarker</code> ada
DaysAfterInitiation	Menentukan jumlah hari setelah memulai upload multipart ketika upload multipart harus diselesaikan. Jika tidak selesai dengan jumlah hari yang ditentukan, itu memenuhi syarat untuk operasi pembatalan dan Amazon S3 membatalkan unggahan multipart yang tidak lengkap. Jenis: Integer Positif Leluhur: <code>AbortIncompleteMultipartUpload</code>	Ya, jika tag induk ditentukan

Nama	Deskripsi	Diperlukan
Expiration	Tindakan ini menentukan periode dalam masa pakai objek saat Amazon S3 harus mengambil tindakan kedaluwarsa yang sesuai. Tindakan yang dilakukan Amazon S3 tergantung pada apakah bucket diaktifkan versi. • Jika pembuatan versi belum pernah diaktifkan pada bucket, Amazon S3 menghapus satu-satunya salinan objek secara permanen. • Jika bucket diaktifkan versi (atau pembuatan versi ditangguhkan), tindakan hanya berlaku untuk versi objek saat ini. Bucket berkemampuan versi dapat memiliki banyak versi dari objek yang sama: satu versi saat ini dan nol atau lebih versi noncurrent. Alih-alih menghapus versi saat ini, Amazon S3 menjadikannya versi noncurrent dengan menambahkan penanda hapus sebagai versi baru saat ini.  Important Jika status bucket ditangguhkan versi, Amazon S3 membuat penanda hapus dengan ID versi. null Jika Anda memiliki versi dengan ID versinull, Amazon S3 akan mengganti versi tersebut.	Ya, jika tidak ada tindakan lain yang hadir diRule.

Nama	Deskripsi	Diperlukan
	 Note Untuk mengatur kedaluwarsa untuk objek noncurrent, gunakan <code>NoncurrentVersionExpiration</code> tindakan. Tipe: Kontainer Anak-anak: Hari atau Tanggal Leluhur: Aturan	
ID	Pengidentifikasi unik untuk aturan. Nilai tidak boleh lebih dari 255 karakter. Tipe: String Leluhur: Aturan	Tidak
LifecycleConfiguration	Wadah untuk aturan siklus hidup. Anda dapat menambahkan sebanyak 1000 aturan. Tipe: Kontainer Anak-anak: Aturan Leluhur: Tidak ada	Ya

Nama	Deskripsi	Diperlukan
ExpiredObjectDeleteMarker	Pada bucket berversi (bucket yang diaktifkan versi atau yang ditangguhkan versi), Anda dapat menambahkan elemen ini dalam konfigurasi siklus hidup untuk memberi tahu Amazon S3 menghapus penanda penghapusan objek yang kedaluwarsa. Sebagai contoh, lihat Contoh 8: Menghapus Penanda Penghapusan Objek Kedaluwarsa di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Jangan menambahkannya ke bucket tanpa versi, karena jenis bucket itu tidak dapat menyertakan penanda hapus. Tipe: String Nilai yang valid: true false (nilainya false diperbolehkan, tetapi tidak ada operasi, yang berarti Amazon S3 tidak akan mengambil tindakan) Leluhur: Expiration	Ya, jika Date dan tidak Days ada

Nama	Deskripsi	Diperlukan
<code>NoncurrentDays</code>	Menentukan jumlah hari suatu obyek menjadi tidak terkini sebelum Amazon S3 dapat melakukan tindakan terkait. Untuk informasi tentang perhitungan hari noncurrent, lihat Cara Amazon S3 Menghitung Saat Obyek Menjadi Tidak Current di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Jenis: Integer Nonnegatif saat digunakan dengan <code>NoncurrentVersionTransition</code>, Integer Positif saat digunakan dengan <code>NoncurrentVersionExpiration</code> Leluhur: <code>NoncurrentVersionExpiration</code> atau <code>NoncurrentVersionTransition</code>	Ya
<code>NoncurrentVersionExpiration</code>	Menentukan kapan versi objek noncurrent kedaluwarsa. Setelah kedaluwarsa, Amazon S3 menghapus versi objek noncurrent secara permanen. Setel tindakan konfigurasi siklus hidup ini pada bucket yang mengaktifkan (atau ditangguhkan) pembuatan versi untuk memberi tahu Amazon S3 agar menghapus versi objek noncurrent pada periode tertentu dalam masa pakai objek. Tipe: Kontainer Anak-anak: <code>NoncurrentDays</code> Leluhur: Aturan	Ya, jika tidak ada tindakan lain yang hadir di Rule

Nama	Deskripsi	Diperlukan
NoncurrentVersionTransition	Wadah untuk aturan transisi yang menjelaskan kapan objek noncurrent transisi ke kelas STANDARD_IA , ONEZONE_IA , atau penyimpanan. Jika bucket Anda mengaktifkan versi (atau jika pembuatan versi ditangguhkan), Anda dapat mengatur tindakan ini untuk memberi tahu Amazon S3 agar melakukan transisi versi objek noncurrent pada periode tertentu dalam masa pakai objek. Tipe: Kontainer Anak-anak: NoncurrentDays dan StorageClass Leluhur: Aturan	Ya, jika tidak ada tindakan lain yang hadir di Rule
Prefix	Awalan kunci objek yang mengidentifikasi satu atau lebih objek yang berlaku aturan. Tipe: String Leluhur: Aturan	Ya
Rule	Wadah untuk aturan siklus hidup. Konfigurasi siklus hidup dapat berisi sebanyak 1000 aturan. Tipe: Kontainer Ancestor: LifecycleConfiguration	Ya

Nama	Deskripsi	Diperlukan
Status	Jika diaktifkan, Amazon S3 mengeksekusi aturan seperti yang dijadwalkan. Jika dinonaktifkan, Amazon S3 mengabaikan aturan tersebut. Tipe: String Leluhur: Aturan Nilai yang valid: Diaktifkan, Dinonaktifkan	Ya
StorageClass	Menentukan kelas penyimpanan Amazon S3 yang Anda inginkan untuk transisi objek. Tipe: String Leluhur: Transisi dan NoncurrentVersionTransition Nilai yang valid: STANDARD_IA ONEZONE_IA GLACIER	Ya Elemen ini diperlukan hanya jika Anda menentukan satu atau kedua leluhurnya.

Nama	Deskripsi	Diperlukan
Transition	Tindakan ini menentukan periode dalam masa pakai objek saat Amazon S3 harus mentransisikannya ke kelas <code>STANDARD_IA</code>, <code>ONEZONE_IA</code>, atau penyimpanan. Ketika tindakan ini berlaku, apa yang dilakukan Amazon S3 bergantung pada apakah bucket diaktifkan versi. • Jika pembuatan versi belum pernah diaktifkan pada bucket, Amazon S3 mengalihkan satu-satunya salinan objek ke kelas penyimpanan yang ditentukan. • Jika bucket Anda mengaktifkan versi (atau pembuatan versi ditangguhkan), Amazon S3 hanya mengalihkan versi objek saat ini yang diidentifikasi dalam aturan.  Note Bucket berkemampuan versi dapat memiliki banyak versi objek. Tindakan ini tidak berpengaruh pada versi objek noncurrent. Untuk transisi objek noncurrent, Anda harus menggunakan <code>NoncurrentVersionTransition</code> tindakan. Tipe: Kontainer	Ya, jika tidak ada tindakan lain yang hadir di Rule

Nama	Deskripsi	Diperlukan
	Anak-anak: Hari atau Tanggal, dan StorageClass	
	Leluhur: Aturan	

Respons

Header Respons

Implementasi operasi ini hanya menggunakan header respons yang umum untuk sebagian besar respons. Untuk informasi selengkapnya, lihat [Header Respons Umum](#).

Elemen Respons

Implementasi operasi ini tidak mengembalikan elemen respons.

Kesalahan Khusus

Implementasi operasi ini tidak mengembalikan kesalahan khusus. Untuk informasi umum tentang kesalahan Amazon S3 dan daftar kode kesalahan, lihat Respons [Kesalahan](#).

Contoh

Contoh 1: Menambahkan Konfigurasi Siklus Hidup ke Bucket yang Tidak Versioning-enabled

Konfigurasi siklus hidup berikut menentukan dua aturan, masing-masing dengan satu tindakan.

- Tindakan Transition memberitahu Amazon S3 untuk melakukan transisi objek dengan awalan “documents/” ke kelas penyimpanan 30 hari setelah pembuatan.
- Tindakan Kedaluwarsa memberitahu Amazon S3 untuk menghapus objek dengan awalan “logs/” 365 hari setelah pembuatan.

```
<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>

```

Berikut ini adalah contoh PUT `/?lifecycle` permintaan yang menambahkan konfigurasi siklus hidup sebelumnya ke `examplebucket` bucket.

```

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>

```

Berikut ini adalah contoh respons.

```
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
```

Contoh 2: Menambahkan Konfigurasi Siklus Hidup ke Versioning-enabled Bucket

Konfigurasi siklus hidup berikut menentukan dua aturan, masing-masing dengan satu tindakan untuk dilakukan Amazon S3. Anda menentukan tindakan ini ketika bucket Anda diaktifkan versi atau pembuatan versi ditangguhkan:

- `T NoncurrentVersionExpiration` indakan memberitahu Amazon S3 untuk mengakhiri versi objek noncurrent dengan awalan “logs/” 100 hari setelah objek menjadi noncurrent.
- `T NoncurrentVersionTransition` indakan memberitahu Amazon S3 untuk mentransisikan versi objek noncurrent dengan awalan “documents/” ke kelas penyimpanan 30 hari setelah objek menjadi noncurrent.

```
<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>
```

Berikut ini adalah contoh PUT `/?lifecycle` permintaan yang menambahkan konfigurasi siklus hidup sebelumnya ke `examplebucket` bucket.

```
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>
```

Berikut ini adalah contoh respons.

```
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
```

Contoh Tambahan

Untuk contoh lebih lanjut tentang transisi objek ke kelas penyimpanan seperti STANDARD_IA atau ONEZONE_IA, lihat Contoh Konfigurasi Siklus Hidup. <https://docs.aws.amazon.com/AmazonS3/latest/userguide/intro-lifecycle-rules.html#lifecycle-configuration-examples>

Sumber Daya Terkait

- [GetBucketLifecycleConfiguration](#)
- [Pemulihan objek POST](#)
- Secara default, pemilik sumber daya—dalam hal ini, pemilik bucket, yang membuat bucket—dapat melakukan operasi apa pun. Akun AWS Pemilik sumber daya juga dapat memberikan izin kepada orang lain untuk melakukan operasi. Untuk informasi selengkapnya, lihat topik berikut di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon:
 - [Menentukan Izin dalam Kebijakan](#)
 - [Mengelola Izin Akses ke Sumber Daya Amazon S3 Anda](#)

Siklus hidup GET Bucket (Tidak digunakan lagi)

Deskripsi

Important

Untuk versi terbaru dari API ini, lihat [GetBucketLifecycleConfiguration](#). Jika Anda mengonfigurasi siklus hidup bucket menggunakan <filter>elemen, Anda akan melihat versi terbaru dari topik ini. Topik ini disediakan untuk kompatibilitas mundur.

Mengembalikan informasi `lifecycle` konfigurasi yang ditetapkan pada bucket. Untuk informasi tentang konfigurasi siklus hidup, buka Manajemen Siklus [Hidup Ob](#) jek di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Untuk menggunakan operasi ini, Anda harus memiliki izin untuk melakukan `s3:GetLifecycleConfiguration` tindakan. Pemilik bucket memiliki izin ini secara default. Pemilik bucket dapat memberikan izin ini kepada orang lain. Untuk informasi selengkapnya tentang izin, lihat M [engelola Izin Akses ke Sumber Daya Amazon S3 Anda](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Permohonan

Sintaksis

```
GET /?lifecycle HTTP/1.1
Host: bucketname.s3.amazonaws.com
Date: date
Authorization: authorization string (see Permintaan Otentikasi \(AWS Versi Tanda Tangan 4\))
```

Parameter Permintaan

Implementasi operasi ini tidak menggunakan parameter permintaan.

Header Permintaan

Implementasi operasi ini hanya menggunakan header permintaan yang umum untuk semua operasi. Untuk informasi selengkapnya, lihat [Header Permintaan Umum](#).

Elemen permintaan

Implementasi operasi ini tidak menggunakan elemen permintaan.

Respons

Header Respons

Implementasi operasi ini hanya menggunakan header respons yang umum untuk sebagian besar respons. Untuk informasi selengkapnya, lihat [Header Respons Umum](#).

Elemen Respons

Implementasi ini GET mengembalikan elemen respons berikut.

Nama	Deskripsi	Diperlukan
AbortIncompleteMultipartUpload	Wadah untuk menentukan kapan unggahan multipart yang tidak lengkap memenuhi syarat untuk operasi pembatalan. Anak: DaysAfterInitiation Tipe: Kontainer Leluhur: Rule	Ya, jika tidak ada tindakan lain yang ditentukan untuk aturan
Date	Tanggal kapan Anda ingin Amazon S3 mengambil tindakan. Untuk informasi selengkapnya, lihat Aturan Siklus Hidup: Berdasarkan Tanggal Ter tentu di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Nilai tanggal harus sesuai dengan format ISO 8601. Waktunya adalah selalu tengah malam UTC. Tipe: String	Ya, jika Days dan tidak ExpiredObjectDeleteMarker ada

Nama	Deskripsi	Diperlukan
	Leluhur: <code>Expiration</code> atau <code>Transition</code>	
Days	Menentukan jumlah hari setelah pembuatan objek ketika tindakan aturan tertentu berlaku. Waktu kelayakan objek dihitung sebagai waktu pembuatan+jumlah hari dengan waktu yang dihasilkan dibulatkan ke tengah malam UTC hari berikutnya. Jenis: Non-negative Integer saat digunakan dengan <code>Transition</code> , Integer Positif saat digunakan dengan <code>Expiration</code> . Leluhur: <code>Transition</code> atau <code>Expiration</code>	Ya, jika <code>Date</code> dan tidak <code>ExpiredObjectDeleteMarker</code> ada
DaysAfterInitiation	Menentukan jumlah hari setelah memulai upload multipart ketika upload multipart harus diselesaikan. Jika tidak selesai dengan jumlah hari yang ditentukan, itu memenuhi syarat untuk operasi pembatalan dan Amazon S3 membatalkan unggahan multipart yang tidak lengkap. Jenis: Integer Positif Leluhur: <code>AbortIncompleteMultipartUpload</code>	Ya, jika <code>Date</code> tidak ada

Nama	Deskripsi	Diperlukan
Expiration	Tindakan ini menentukan periode dalam masa pakai objek saat Amazon S3 harus mengambil tindakan kedaluwarsa yang sesuai. Tindakan kedaluwarsa hanya terjadi pada objek yang memenuhi syarat sesuai dengan periode yang ditentukan dalam anak <code>Date</code> atau <code>Days</code> elemen. Tindakan yang dilakukan Amazon S3 tergantung pada apakah bucket diaktifkan pembuatan versi. • Jika pembuatan versi belum pernah diaktifkan pada bucket, Amazon S3 menghapus satu-satunya salinan objek secara permanen. • Jika tidak, jika bucket Anda mengaktifkan versi (atau pembuatan versi ditangguhkan), tindakan hanya berlaku untuk versi objek saat ini. Bucket yang diaktifkan versi atau ditangguhkan versi dapat memiliki banyak versi dari objek yang sama: satu versi saat ini, dan nol atau lebih versi noncurrent. Alih-alih menghapus versi saat ini, Amazon S3 menjadikannya versi noncurrent dengan menambahkan penanda hapus sebagai versi baru saat ini.  Important Jika status bucket ditangguhkan versi, Amazon S3 membuat penanda hapus dengan ID versi. <code>null</code> Jika	Ya, jika tag induk ditentukan

Nama	Deskripsi	Diperlukan
	Anda memiliki versi dengan ID <code>versinu11</code>, maka Amazon S3 akan mengganti versi tersebut.  Note Untuk mengatur kedaluwarsa untuk objek <code>noncurrent</code>, Anda harus menggunakan <code>NoncurrentVersionExpiration</code> tindakan. Tipe: Kontainer Anak-anak: Hari atau Tanggal Leluhur: Aturan	
ID	Pengidentifikasi unik untuk aturan. Nilai tidak boleh lebih dari 255 karakter. Tipe: String Leluhur: Aturan	Tidak

Nama	Deskripsi	Diperlukan
LifecycleConfiguration	Wadah untuk aturan siklus hidup. Anda dapat menambahkan sebanyak 1000 aturan. Tipe: Kontainer Anak-anak: Aturan Leluhur: Tidak ada	Ya
ExpiredObjectDeleteMarker	Pada bucket berversi (bucket versioning-enabled atau versioning-bucket), elemen ini menunjukkan apakah Amazon S3 akan menghapus penanda penghapusan objek yang kedaluwarsa di bucket. Sebagai contoh, buka Contoh 8: Tentukan Tindakan Kedaluwarsa untuk Menghapus Penanda Penghapusan Objek Kedaluwarsa di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Tipe: String Nilai yang valid: true false (nilai false diperbolehkan tetapi tidak ada operasi, Amazon S3 tidak mengambil tindakan jika nilainya salah) Leluhur: Expiration	Ya, jika Date dan tidak Days ada

Nama	Deskripsi	Diperlukan
NoncurrentDays	Menentukan jumlah hari suatu objek tidak berjalan sebelum Amazon S3 dapat melakukan tindakan terkait. Untuk informasi tentang menghitung hari noncurrent, lihat Aturan Siklus Hidup Berdasarkan Jumlah Hari di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon. Jenis: Integer Nonnegatif saat digunakan denganNoncurrentVersionTransition , Integer Positif saat digunakan dengan NoncurrentVersionExpiration Leluhur: NoncurrentVersionExpiration atau NoncurrentVersionTransition	Ya, hanya jika leluhur hadir
NoncurrentVersionExpiration	Menentukan kapan versi objek noncurrent kedaluwarsa. Setelah kedaluwarsa, Amazon S3 menghapus versi objek noncurrent secara permanen. Tetapkan tindakan konfigurasi siklus hidup ini pada bucket yang mengaktifkan (atau ditanggguhkan) pembuatan versi untuk meminta agar Amazon S3 menghapus versi objek noncurrent pada periode tertentu dalam masa pakai objek. Tipe: Kontainer Anak-anak: NoncurrentDays Leluhur: Aturan	Ya, jika tidak ada tindakan lain yang hadir di Rule

Nama	Deskripsi	Diperlukan
NoncurrentVersionTransition	Wadah untuk aturan transisi yang menjelaskan kapan objek noncurrent transisi ke STANDARD_IA , ONEZONE_IA , atau kelas penyimpanan. Jika bucket Anda mengaktifkan versi (atau pembuatan versi ditangguhkan), Anda dapat mengatur tindakan ini untuk meminta Amazon S3 untuk mentransisi versi objek noncurrent ke kelas penyimpanan pada periode tertentu dalam masa pakai objek. Tipe: Kontainer Anak-anak: NoncurrentDays dan StorageClass Leluhur: Aturan	Ya, jika tidak ada tindakan lain yang hadir di Rule
Prefix	Awalan kunci objek mengidentifikasi satu atau lebih objek yang berlaku aturan. Tipe: String Leluhur: Aturan	Ya
Rule	Wadah untuk aturan siklus hidup. Tipe: Kontainer Leluhur: LifecycleConfiguration	Ya

Nama	Deskripsi	Diperlukan
Status	Jika Diaktifkan, Amazon S3 mengeksekusi aturan seperti yang dijadwalkan. Jika Dinonaktifkan, Amazon S3 mengabaikan aturan tersebut. Tipe: String Leluhur: Aturan Nilai yang valid: Diaktifkan atau Dinonaktifkan	Ya
StorageClass	Menentukan kelas penyimpanan Amazon S3 yang ingin Anda transisi objek. Tipe: String Leluhur: Transisi dan NoncurrentVersionTransition Nilai yang valid: STANDARD_IA ONEZONE_IA	Ya

Nama	Deskripsi	Diperlukan
Transition	Tindakan ini menentukan periode dalam masa pakai objek saat Amazon S3 harus mentransisikannya ke kelas <code>STANDARD_IA</code>, <code>ONEZONE_IA</code>, atau penyimpanan. Ketika tindakan ini berlaku, apa yang dilakukan Amazon S3 bergantung pada apakah bucket diaktifkan versi. • Jika pembuatan versi belum pernah diaktifkan pada bucket, Amazon S3 mengalihkan satu-satunya salinan objek ke kelas penyimpanan yang ditentukan. • Saat bucket Anda mengaktifkan versi (atau pembuatan versi ditangguhkan), Amazon S3 hanya mentransisi versi terkini dari objek yang diidentifikasi dalam aturan.  Note Bucket yang diaktifkan versi atau suspensi versi dapat berisi banyak versi objek. Tindakan ini tidak berpengaruh pada versi objek noncurrent. Untuk transisi objek noncurrent, Anda harus menggunakan <code>NoncurrentVersionTransition</code> tindakan. Tipe: Kontainer	Ya, jika tidak ada tindakan lain yang hadir di Rule

Nama	Deskripsi	Diperlukan
	Anak-anak: Hari atau Tanggal, dan StorageClass	
	Leluhur: Aturan	

Kesalahan Khusus

Kode Kesalahan	Deskripsi	Kode Status HTTP	Awalan Kode Kesalahan SOAP
NoSuchLifecycleConfiguration	Konfigurasi siklus hidup tidak ada.	404 Tidak Ditemukan	Klien

Untuk informasi umum tentang kesalahan Amazon S3 dan daftar kode kesalahan, lihat [Tanggapan kesalahan](#).

Contoh

Contoh 1: Mengambil Sub-Sumber Daya Siklus Hidup

Contoh ini adalah permintaan GET untuk mengambil lifecycle subresource dari bucket yang ditentukan, dan contoh respons dengan konfigurasi siklus hidup yang dikembalikan.

Permintaan Sampel

```
GET /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Thu, 15 Nov 2012 00:17:21 GMT
Authorization: signatureValue
```

Contoh Respons

```
HTTP/1.1 200 OK
x-amz-id-2: ITnGT1y4RyTmXa3rPi4hklTXouTf0hccUjo0iCPjz6FnfIutBj3M7fPG1W02SEWp
```



```
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>
```

Sumber Daya Terkait

- [PutBucketLifecycleConfiguration](#)
- [DeleteBucketLifecycle](#)

Riwayat dokumen

Tabel berikut menjelaskan perubahan penting pada Panduan Pengembang Amazon S3. Untuk notifikasi tentang pembaruan dokumentasi ini, Anda dapat berlangganan ke umpan RSS.

Perubahan	Deskripsi	Tanggal
Panduan Pengembang Baru	Memindahkan semua konten non-API, seperti pengembangan dengan S3, contoh kode, respons kesalahan, dan lainnya dari panduan referensi API Amazon S3 ke panduan pengembang Amazon S3 baru ini. Untuk informasi selengkapnya, lihat Referensi API Amazon S3 .	Juni 15, 2026

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.