



Entwicklerhandbuch

Amazon Elastic Compute Cloud



Amazon Elastic Compute Cloud: Entwicklerhandbuch

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Die Marken und Handelsmarken von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, die geeignet ist, Kunden irrezuführen oder Amazon in irgendeiner Weise herabzusetzen oder zu diskreditieren. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Programmatischer Zugriff auf Amazon EC2	1
Service-Endpunkte	1
IPv4Endpunkte	2
Dual-Stack IPv4 - (undIPv6) Endpunkte	2
Service-Endpunkte nach Regionen	3
Angaben von Endpunkten	7
Letztendliche Datenkonsistenz	9
Idempotenz	11
Idempotenz bei Amazon EC2	12
RunInstances Idempotenz	15
Beispiele	17
Versuchen Sie es erneut mit Empfehlungen für idempotente Anfragen	18
API-Drosselung beantragen	19
Wie wird die Drosselung angewendet	20
Drosselungsgrenzwerte	22
Drosselung überwachen API	35
Wiederholungen und exponentieller Backoff	35
Anfordern einer -Limit-Erhöhung	36
Mit dem AWS CLI	38
Erfahren Sie mehr über AWS CLI	38
Verwenden AWS CloudFormation	39
Amazon EC2 und Vorlagen AWS CloudFormation	39
Ressourcen für Amazon EC2	39
Erfahre mehr über AWS CloudFormation	43
Mit einem AWS SDK	44
Codebeispiele für den Amazon EC2 API	44
Erfahren Sie mehr über die AWS SDKs	44
Low-Level-API für Amazon EC2	45
Console-to-Code	46
Codebeispiele	47
Grundlagen	67
Hallo Amazon EC2	74
Erlernen Sie die Grundlagen	87
Aktionen	304

Szenarien	1058
Erstellen und Verwalten eines ausfallsicheren Services	1059
Überwachen Sie API Anfragen mit CloudWatch	1228
EC2APIAmazon-Metriken aktivieren	1228
EC2APIAmazon-Metriken und -Dimensionen	1229
Metriken	1229
Dimensionen	1230
Aufbewahrung metrischer Daten	1231
Überwachung von Anfragen, die in Ihrem Namen gestellt wurden	1231
Fakturierung	1231
Mit Amazon arbeiten CloudWatch	1231
CloudWatch Metriken anzeigen	1231
CloudWatch Alarme erstellen	1232
.....	mccxxxv

Programmatischer Zugriff auf Amazon EC2

Sie können Ihre Amazon EC2 EC2-Ressourcen mithilfe der AWS Management Console oder einer programmatischen Schnittstelle erstellen und verwalten. Informationen zur Verwendung der Amazon EC2 EC2-Konsole finden Sie im [Amazon EC2 EC2-Benutzerhandbuch](#).

Funktionsweise

- [Amazon EC2 EC2-Endpunkte](#)
- [Eventuelle Konsistenz](#)
- [Ich bin Impotenz](#)
- [Drosselung beantragen](#)

Programmierschnittstellen

- [AWS Command Line Interface \(AWS CLI\)](#)
- [AWS CloudFormation](#)
- [AWS SDKs](#)
- [API auf niedriger Ebene](#)

Erste Schritte

- [Codebeispiele](#)
- [Console-to-Code](#)

Überwachen

- [AWS CloudTrail](#)
- [Anfragen überwachen](#)

EC2Amazon-Serviceendpunkte

Ein Endpunkt ist ein EndpunktURL, der als Einstiegspunkt für einen AWS Webdienst dient. Amazon EC2 unterstützt die folgenden Endpunkttypen:

- [IPv4-Endpunkte](#)
- [Dual-Stack-Endpunkte](#) (unterstützen sowohl als auch) IPv4 IPv6
- [FIPSEndpunkte](#)

Wenn Sie eine Anfrage stellen, können Sie den zu verwendenden Endpunkt angeben. Wenn Sie keinen Endpunkt angeben, wird der IPv4 Endpunkt standardmäßig verwendet. Um einen anderen Endpunkttyp zu verwenden, müssen Sie ihn in Ihrer Anforderung angeben. Beispiele für diese Vorgehensweise finden Sie unter [Angeben von Endpunkten](#). Eine Tabelle der verfügbaren Endpunkte finden Sie unter [Service-Endpunkte nach Regionen](#).

IPv4-Endpunkte

IPv4-Endpunkte unterstützen nur IPv4 Datenverkehr. IPv4-Endpunkte sind für alle Regionen verfügbar.

Wenn Sie den allgemeinen Endpunkt, `ec2.amazonaws.com`, angeben, verwenden wir den Endpunkt für `us-east-1`. Um eine andere Region zu verwenden, geben Sie den zugehörigen Endpunkt an. Wenn Sie beispielsweise `ec2.us-east-2.amazonaws.com` als Endpunkt angeben, leiten wir Ihre Anfrage an den `us-east-2`-Endpunkt weiter.

IPv4-Endpunktnamen verwenden die folgende Benennungskonvention:

- `service.region.amazonaws.com`

Der IPv4 Endpunktname für die `eu-west-1` Region lautet beispielsweise `ec2.eu-west-1.amazonaws.com`.

Dual-Stack IPv4 - (und IPv6) Endpunkte

Dual-Stack-Endpunkte unterstützen sowohl den als auch den Datenverkehr. IPv4 IPv6 Wenn Sie eine Anfrage an einen Dual-Stack-Endpunkt stellen, wird der Endpunkt je nach dem von Ihrem Netzwerk und Client verwendeten Protokoll in eine IPv6 oder eine IPv4 Adresse URL aufgelöst.

Amazon EC2 unterstützt nur regionale Dual-Stack-Endpunkte, was bedeutet, dass Sie die Region als Teil des Endpunktnamens angeben müssen. Dual-Stack-Endpunktnamen verwenden die folgende Namenskonvention:

- `ec2.region.api.aws`

Beispielsweise ist der Dual-Stack-Endpunktname für die Region eu-west-1 `ec2.eu-west-1.api.aws`.

Service-Endpunkte nach Regionen

Im Folgenden sind die Service-Endpunkte für Amazon EC2 aufgeführt. Weitere Informationen zu Regionen finden Sie unter [Regionen und Availability Zones](#) im EC2Amazon-Benutzerhandbuch.

Name der Region	Region	Endpunkt	Protocol (Protokoll)
USA Ost (Ohio)	us-east-2	<code>ec2.us-east-2.amazonaws.com</code>	HTTP und HTTPS
		<code>ec2-fips.us-east-2.amazonaws.com</code>	HTTPS
		<code>ec2.us-east-2.api.aws</code>	HTTPS
USA Ost (Nord-Virginia)	us-east-1	<code>ec2.us-east-1.amazonaws.com</code>	HTTP und HTTPS
		<code>ec2-fips.us-east-1.amazonaws.com</code>	HTTPS
		<code>ec2.us-east-1.api.aws</code>	HTTPS
USA West (Nordkalifornien)	us-west-1	<code>ec2.us-west-1.amazonaws.com</code>	HTTP und HTTPS
		<code>ec2-fips.us-west-1.amazonaws.com</code>	HTTPS
		<code>ec2.us-west-1.api.aws</code>	HTTPS
USA West (Oregon)	us-west-2	<code>ec2.us-west-2.amazonaws.com</code>	HTTP und HTTPS
		<code>ec2-fips.us-west-2.amazonaws.com</code>	HTTPS
		<code>ec2.us-west-2.api.aws</code>	HTTPS

Name der Region	Region	Endpoint	Protocol (Protokoll)
Afrika (Kapstadt)	af-south-1	ec2.af-south-1.amazonaws.com ec2.af-south-1.api.aws	HTTPund HTTPS HTTPS
Asien-Pazifik (Hongkong)	ap-east-1	ec2.ap-east-1.amazonaws.com ec2.ap-east-1.api.aws	HTTPund HTTPS HTTPS
Asien-Pazifik (Hyderabad)	ap-south-2	ec2.ap-south-2.amazonaws.com	HTTPS
Asien-Pazifik (Jakarta)	ap-southeast-3	ec2.ap-southeast-3.amazonaws.com	HTTPS
Asien-Pazifik (Malaysia)	ap-southeast-5	ec2.ap-southeast-5.amazonaws.com	HTTPS
Asien-Pazifik (Melbourne)	ap-southeast-4	ec2.ap-southeast-4.amazonaws.com	HTTPS
Asien-Pazifik (Mumbai)	ap-south-1	ec2.ap-south-1.amazonaws.com ec2.ap-south-1.api.aws	HTTPund HTTPS HTTPS

Name der Region	Region	Endpoint	Protocol (Protokoll)
Asien-Pazifik (Osaka)	ap-northeast-3	ec2.ap-northeast-3.amazonaws.com	HTTP und HTTPS
Asien-Pazifik (Seoul)	ap-northeast-2	ec2.ap-northeast-2.amazonaws.com ec2.ap-northeast-2.api.aws	HTTP und HTTPS HTTPS
Asien-Pazifik (Singapur)	ap-southeast-1	ec2.ap-southeast-1.amazonaws.com ec2.ap-southeast-1.api.aws	HTTP und HTTPS HTTPS
Asien-Pazifik (Sydney)	ap-southeast-2	ec2.ap-southeast-2.amazonaws.com ec2.ap-southeast-2.api.aws	HTTP und HTTPS HTTPS
Asien-Pazifik (Tokio)	ap-northeast-1	ec2.ap-northeast-1.amazonaws.com ec2.ap-northeast-1.api.aws	HTTP und HTTPS HTTPS
Kanada (Zentral)	ca-central-1	ec2.ca-central-1.amazonaws.com ec2-fips.ca-central-1.amazonaws.com ec2.ca-central-1.api.aws	HTTP und HTTPS HTTPS HTTPS
Kanada West (Calgary)	ca-west-1	ec2.ca-west-1.amazonaws.com ec2-fips.ca-west-1.amazonaws.com	HTTPS HTTPS

Name der Region	Region	Endpoint	Protocol (Protokoll)
Europa (Frankfurt)	eu-central-1	ec2.eu-central-1.amazonaws.com ec2.eu-central-1.api.aws	HTTPund HTTPS HTTPS
Europa (Irland)	eu-west-1	ec2.eu-west-1.amazonaws.com ec2.eu-west-1.api.aws	HTTPund HTTPS HTTPS
Europa (London)	eu-west-2	ec2.eu-west-2.amazonaws.com ec2.eu-west-2.api.aws	HTTPund HTTPS HTTPS
Europa (Mailand)	eu-south-1	ec2.eu-south-1.amazonaws.com ec2.eu-south-1.api.aws	HTTPund HTTPS HTTPS
Europa (Paris)	eu-west-3	ec2.eu-west-3.amazonaws.com ec2.eu-west-3.api.aws	HTTPund HTTPS HTTPS
Europa (Spanien)	eu-south-2	ec2.eu-south-2.amazonaws.com	HTTPS
Europa (Stockholm)	eu-north-1	ec2.eu-north-1.amazonaws.com ec2.eu-north-1.api.aws	HTTPund HTTPS HTTPS
Europa (Zürich)	eu-central-2	ec2.eu-central-2.amazonaws.com	HTTPS

Name der Region	Region	Endpunkt	Protocol (Protokoll)
Israel (Tel Aviv)	il-centra l-1	ec2.il-central-1.amazonaws.com	HTTPS
Naher Osten (Bahrain)	me- south-1	ec2.me-south-1.amazonaws.com ec2.me-south-1.api.aws	HTTPund HTTPS HTTPS
Naher Osten (UAE)	me- central-1	ec2.me-central-1.amazonaws.com	HTTPS
Südamerika (São Paulo)	sa-east-1	ec2.sa-east-1.amazonaws.com ec2.sa-east-1.api.aws	HTTPund HTTPS HTTPS
AWS GovCloud (US-Ost)	us-gov- east-1	ec2.us-gov-east-1.amazonaws.com ec2.us-gov-east-1.api.aws	HTTPS HTTPS
AWS GovCloud (US-West)	us-gov- west-1	ec2.us-gov-west-1.amazonaws.com ec2.us-gov-west-1.api.aws	HTTPS HTTPS

Angeben von Endpunkten

Dieser Abschnitt enthält einige Beispiele dafür, wie Sie einen Endpunkt angeben, wenn Sie eine Anforderung stellen.

AWS CLI

Die folgenden Beispiele zeigen, wie Sie mithilfe von einen Endpunkt für die us-east-2 Region angeben AWS CLI.

- Dual-Stack

```
aws ec2 describe-regions --region us-east-2 --endpoint-url https://ec2.us-east-2.api.aws
```

- IPv4

```
aws ec2 describe-regions --region us-east-2 --endpoint-url https://ec2.us-east-2.amazonaws.com
```

AWS SDK for Java 2.x

Die folgenden Beispiele zeigen, wie Sie mithilfe von einem Endpunkt für die `us-east-2` Region angeben AWS SDK for Java 2.x.

- Dual-Stack

```
Ec2Client client = Ec2Client.builder()  
    .region(Region.US_EAST_2)  
    .endpointOverride(URI.create("https://ec2.us-east-2.api.aws"))  
    .build();
```

- IPv4

```
Ec2Client client = Ec2Client.builder()  
    .region(Region.US_EAST_2)  
    .endpointOverride(URI.create("https://ec2.us-east-2.amazonaws.com"))  
    .build();
```

AWS SDK for Java 1.x

Die folgenden Beispiele zeigen, wie Sie mithilfe von AWS SDK for Java 1.x einen Endpunkt für die `eu-west-1` Region angeben.

- Dual-Stack

```
AmazonEC2 s3 = AmazonEC2ClientBuilder.standard()  
    .withEndpointConfiguration(new EndpointConfiguration(  
        "https://ec2.eu-west-1.api.aws",  
        "eu-west-1"))
```

```
.build();
```

- IPv4

```
AmazonEC2 s3 = AmazonEC2ClientBuilder.standard()
    .withEndpointConfiguration(new EndpointConfiguration(
        "https://ec2.eu-west-1.amazonaws.com",
        "eu-west-1"))
    .build();
```

AWS SDK for Go

Die folgenden Beispiele zeigen, wie Sie mit dem einen Endpunkt für die us-east-1 Region angeben. AWS SDK for Go

- Dual-Stack

```
sess := session.Must(session.NewSession())
svc := ec2.New(sess, &aws.Config{
    Region: aws.String(endpoints.UsEast1RegionID),
    Endpoint: aws.String("https://ec2.us-east-1.api.aws")
})
```

- IPv4

```
sess := session.Must(session.NewSession())
svc := ec2.New(sess, &aws.Config{
    Region: aws.String(endpoints.UsEast1RegionID),
    Endpoint: aws.String("https://ec2.us-east-1.amazonaws.com")
})
```

Eventuelle Konsistenz im Amazon EC2 API

Amazon EC2 API folgt aufgrund des verteilten Charakters des Systems, das es unterstützt, einem eventuellen Konsistenzmodell. Das bedeutet, dass das Ergebnis eines von Ihnen API ausgeführten Befehls, der sich auf Ihre EC2 Amazon-Ressourcen auswirkt, möglicherweise nicht sofort für alle nachfolgenden Befehle sichtbar ist, die Sie ausführen. Sie sollten dies berücksichtigen, wenn Sie einen API Befehl ausführen, der unmittelbar auf einen vorherigen API Befehl folgt.

Eventuelle Konsistenz kann sich auf die Art und Weise auswirken, wie Sie Ihre Ressourcen verwalten. Wenn Sie beispielsweise einen Befehl ausführen, um eine Ressource zu erstellen, wird diese irgendwann für andere Befehle sichtbar sein. Wenn Sie also einen Befehl ausführen, um die Ressource, die Sie gerade erstellt haben, zu ändern oder zu beschreiben, hat sich ihre ID möglicherweise nicht im gesamten System verbreitet und Sie erhalten eine Fehlermeldung, dass die Ressource nicht existiert.

Um die eventuelle Konsistenz zu gewährleisten, können Sie wie folgt vorgehen:

- Bestätigen Sie den Status der Ressource, bevor Sie einen Befehl ausführen, um sie zu ändern. Führen Sie den entsprechenden `Describe` Befehl mit einem exponentiellen Backoff-Algorithmus aus, um sicherzustellen, dass Sie genügend Zeit für die Ausbreitung des vorherigen Befehls im System einplanen. Führen Sie dazu den `Describe` Befehl wiederholt aus, wobei Sie mit einer Wartezeit von einigen Sekunden beginnen und die Wartezeit schrittweise auf bis zu fünf Minuten erhöhen.
- Fügen Sie Wartezeiten zwischen aufeinanderfolgenden Befehlen hinzu, auch wenn ein `Describe` Befehl eine genaue Antwort zurückgibt. Wenden Sie einen exponentiellen Backoff-Algorithmus an, der mit einer Wartezeit von einigen Sekunden beginnt, und erhöhen Sie die Wartezeit schrittweise auf etwa fünf Minuten.

Beispiele für eventuelle Konsistenzfehler

Im Folgenden finden Sie Beispiele für Fehlercodes, auf die Sie aufgrund einer eventuellen Konsistenz stoßen können.

- `InvalidInstanceID.NotFound`

Wenn Sie den `RunInstances` Befehl erfolgreich ausführen und anschließend sofort einen weiteren Befehl mit der Instanz-ID ausführen, die in der Antwort von angegeben wurde `RunInstances`, wird möglicherweise ein `InvalidInstanceID.NotFound` Fehler zurückgegeben. Das bedeutet nicht, dass die Instanz nicht existiert.

Einige spezifische Befehle, die davon betroffen sein könnten, sind:

- `DescribeInstances`: Um den tatsächlichen Status der Instanz zu überprüfen, führen Sie diesen Befehl mit einem exponentiellen Backoff-Algorithmus aus.
- `TerminateInstances`: Um den Status der Instanz zu bestätigen, führen Sie zunächst den `DescribeInstances` Befehl mit einem exponentiellen Backoff-Algorithmus aus.

⚠ Important

Wenn Sie nach dem Ausführen eine `InvalidInstanceID.NotFound` Fehlermeldung erhalten `TerminateInstances`, bedeutet dies nicht, dass die Instanz beendet ist oder beendet wird. Ihre Instance könnte immer noch laufen. Aus diesem Grund ist es wichtig, zunächst den Status der Instanz mithilfe von `DescribeInstances` zu überprüfen.

- `InvalidGroup.NotFound`

Wenn Sie den `CreateSecurityGroup` Befehl erfolgreich ausführen und dann sofort einen weiteren Befehl mit der Sicherheitsgruppen-ID ausführen, die in der Antwort von `createSecurityGroup` angegeben wurde, wird möglicherweise ein `InvalidGroup.NotFound` Fehler zurückgegeben. Um den Status der Sicherheitsgruppe zu überprüfen, führen Sie den `DescribeSecurityGroups` Befehl mit einem exponentiellen Backoff-Algorithmus aus.

- `InstanceLimitExceeded`

Sie haben mehr Instances angefordert, als Ihr aktuelles Instance-Limit für den angegebenen Instance-Typ zulässt. Sie könnten dieses Limit unerwartet erreichen, wenn Sie Instances schnell starten und beenden, da beendete Instances nach ihrer Kündigung noch eine Weile auf Ihr Instance-Limit angerechnet werden.

Sicherstellung der Idempotenz bei Amazon-Anfragen EC2 API

Wenn Sie eine mutierende API Anfrage stellen, gibt die Anfrage in der Regel ein Ergebnis zurück, bevor die asynchronen Workflows des Vorgangs abgeschlossen sind. Es können auch ein Timeout oder andere Serverprobleme auftreten, bevor Operationen abgeschlossen sind, obwohl die Anfrage bereits ein Ergebnis zurückgegeben hat. Dadurch lässt sich möglicherweise nur schwer feststellen, ob die Anfrage erfolgreich war oder nicht, und es werden möglicherweise mehrere Wiederholungsversuche vorgenommen, um sicherzustellen, dass die Operation erfolgreich abgeschlossen wird. Wenn die ursprüngliche Anfrage und die nachfolgenden Wiederholungsversuche jedoch erfolgreich sind, wird die Operation mehrmals abgeschlossen. Das bedeutet, dass Sie möglicherweise mehr Ressourcen erstellen, als Sie beabsichtigt haben.

Idempotenz stellt sicher, dass eine API Anfrage nicht öfter als einmal abgeschlossen wird. Wenn bei einer idempotenten Anfrage die ursprüngliche Anfrage erfolgreich abgeschlossen wird, werden alle nachfolgenden Wiederholungen erfolgreich abgeschlossen, ohne dass weitere Aktionen ausgeführt

werden. Das Ergebnis kann jedoch aktualisierte Informationen enthalten, z. B. den aktuellen Erstellungsstatus.

Inhalt

- [Idempotenz bei Amazon EC2](#)
- [RunInstances Idempotenz](#)
- [Beispiele](#)
- [Versuchen Sie es erneut mit Empfehlungen für idempotente Anfragen](#)

Idempotenz bei Amazon EC2

Die folgenden API Aktionen sind standardmäßig idempotent und erfordern keine zusätzliche Konfiguration. Die entsprechenden AWS CLI Befehle unterstützen standardmäßig auch Idempotenz.

Standardmäßig ist Idempotent

- AssociateAddress
- CreateVpnConnection
- DisassociateAddress
- ReplaceNetworkAclAssociation
- TerminateInstances

Die folgenden API Aktionen unterstützen optional Idempotenz mithilfe eines Client-Tokens. Die entsprechenden AWS CLI Befehle unterstützen auch Idempotenz mithilfe eines Client-Tokens. Ein Client-Token ist eine eindeutige Zeichenfolge mit bis zu 64 Zeichen, bei der Groß- und Kleinschreibung beachtet wird. ASCII Um mit einer dieser Aktionen eine idempotente API Anfrage zu stellen, geben Sie in der Anfrage ein Client-Token an. Sie sollten dasselbe Client-Token nicht für andere API Anfragen wiederverwenden. Wenn Sie eine Anfrage, die erfolgreich abgeschlossen wurde, mit demselben Client-Token und denselben Parametern erneut versuchen, ist die Wiederholung erfolgreich, ohne dass weitere Aktionen ausgeführt werden. Wenn Sie eine erfolgreiche Anfrage mit demselben Client-Token erneut versuchen, aber einer oder mehrere der Parameter anders sind als die Region oder Availability Zone, schlägt die Wiederholung mit einem Fehler fehl. `IdempotentParameterMismatch`

Ich bin impotent, wenn ein Client-Token verwendet wird

- AllocateHosts
- AllocateIpamPoolCidr
- AssociateClientVpnTargetNetwork
- AssociateIpamResourceDiscovery
- AttachVerifiedAccessTrustProvider
- AuthorizeClientVpnIngress
- CopyFpgaImage
- CopyImage
- CreateCapacityReservation
- CreateCapacityReservationFleet
- CreateClientVpnEndpoint
- CreateClientVpnRoute
- CreateEgressOnlyInternetGateway
- CreateFleet
- CreateFlowLogs
- CreateFpgaImage
- CreateInstanceConnectEndpoint
- CreateIpam
- CreateIpamPool
- CreateIpamResourceDiscovery
- CreateIpamScope
- CreateLaunchTemplate
- CreateLaunchTemplateVersion
- CreateManagedPrefixList
- CreateNatGateway
- CreateNetworkAcl
- CreateNetworkInsightsAccessScope
- CreateNetworkInsightsPath

- `CreateNetworkInterface`
- `CreateReplaceRootVolumeTask`
- `CreateReservedInstancesListing`
- `CreateRouteTable`
- `CreateTrafficMirrorFilter`
- `CreateTrafficMirrorFilterRule`
- `CreateTrafficMirrorSession`
- `CreateTrafficMirrorTarget`
- `CreateVerifiedAccessEndpoint`
- `CreateVerifiedAccessGroup`
- `CreateVerifiedAccessInstance`
- `CreateVerifiedAccessTrustProvider`
- `CreateVolume`
- `CreateVpcEndpoint`
- `CreateVpcEndpointConnectionNotification`
- `CreateVpcEndpointServiceConfiguration`
- `DeleteVerifiedAccessEndpoint`
- `DeleteVerifiedAccessGroup`
- `DeleteVerifiedAccessInstance`
- `DeleteVerifiedAccessTrustProvider`
- `DetachVerifiedAccessTrustProvider`
- `ExportImage`
- `ImportImage`
- `ImportSnapshot`
- `ModifyInstanceCreditSpecification`
- `ModifyLaunchTemplate`
- `ModifyReservedInstances`
- `ModifyVerifiedAccessEndpoint`
- `ModifyVerifiedAccessEndpointPolicy`
- `ModifyVerifiedAccessGroup`

- `ModifyVerifiedAccessGroupPolicy`
- `ModifyVerifiedAccessInstance`
- `ModifyVerifiedAccessInstanceLoggingConfiguration`
- `ModifyVerifiedAccessTrustProvider`
- `ProvisionIpamPoolCidr`
- `PurchaseHostReservation`
- `RequestSpotFleet`
- `RequestSpotInstances`
- `RunInstances`
- `StartNetworkInsightsAccessScopeAnalysis`
- `StartNetworkInsightsAnalysis`

Arten von Idempotenz

- **Regional** — Anfragen sind in jeder Region idempotent. Sie können jedoch dieselbe Anfrage, einschließlich desselben Client-Tokens, in einer anderen Region verwenden.
- **Zonal** — Anfragen sind in jeder Availability Zone in einer Region idempotent. Wenn Sie beispielsweise dasselbe Client-Token in zwei Aufrufen in derselben Region angeben, sind die Aufrufe erfolgreich, wenn sie unterschiedliche Werte für den Parameter angeben. `AllocateHosts`
`AvailabilityZone`

RunInstances Idempotenz

Die [RunInstances](#) API Aktion nutzt sowohl regionale als auch zonale Idempotenz.

Die Art der verwendeten Idempotenz hängt davon ab, wie Sie die Availability Zone in Ihrer Anfrage angeben. RunInstances API Die Anfrage verwendet in den folgenden Fällen zonale Idempotenz:

- Wenn Sie mithilfe des `AvailabilityZoneParameters` im Datentyp `Placement` explizit eine Availability Zone angeben
- Wenn Sie mithilfe des Parameters implizit eine Availability Zone angeben `SubnetId`

Wenn Sie keine Availability Zone explizit oder implizit angeben, verwendet die Anfrage regionale Idempotenz.

Zonale Idempotenz

Die zonale Idempotenz stellt sicher, dass eine RunInstances API Anfrage in jeder Availability Zone in einer Region idempotent ist. Dadurch wird sichergestellt, dass eine Anfrage mit demselben Client-Token innerhalb jeder Availability Zone in einer Region nur einmal abgeschlossen werden kann. Dasselbe Client-Token kann jedoch verwendet werden, um Instances in anderen Availability Zones in der Region zu starten.

Wenn Sie beispielsweise eine idempotente Anfrage senden, um eine Instance in der `us-east-1a` Availability Zone zu starten, und dann dasselbe Client-Token in einer Anfrage in der `us-east-1b` Availability Zone verwenden, starten wir Instances in jeder dieser Availability Zones. Wenn einer oder mehrere der Parameter unterschiedlich sind, kehren nachfolgende Wiederholungen mit demselben Client-Token in diesen Availability Zones entweder erfolgreich zurück, ohne dass weitere Aktionen ausgeführt werden, oder schlagen mit einem Fehler fehl. `IdempotentParameterMismatch`

Regionale Idempotenz

Regionale Idempotenz stellt sicher, dass eine RunInstances API Anfrage in einer Region idempotent ist. Dadurch wird sichergestellt, dass eine Anfrage mit demselben Client-Token innerhalb einer Region nur einmal abgeschlossen werden kann. Genau dieselbe Anfrage mit demselben Client-Token kann jedoch verwendet werden, um Instances in einer anderen Region zu starten.

Wenn Sie beispielsweise eine idempotente Anfrage senden, um eine Instance in der `us-east-1` Region zu starten, und dann dasselbe Client-Token in einer Anfrage in der `eu-west-1` Region verwenden, starten wir Instances in jeder dieser Regionen. Wenn einer oder mehrere der Parameter unterschiedlich sind, kehren nachfolgende Wiederholungen mit demselben Client-Token in diesen Regionen entweder erfolgreich zurück, ohne dass weitere Aktionen ausgeführt werden, oder schlagen mit einem Fehler fehl. `IdempotentParameterMismatch`

Tip

Wenn eine der Availability Zones in der angeforderten Region nicht verfügbar ist, können RunInstances Anfragen, die regionale Idempotenz verwenden, fehlschlagen. Um die von der AWS Infrastruktur angebotenen Availability Zone-Funktionen nutzen zu können, empfehlen wir, beim Starten von Instances die zonale Idempotenz zu verwenden. RunInstances Anfragen, die zonale Idempotenz verwenden und auf eine verfügbare Availability Zone abzielen, sind erfolgreich, auch wenn keine andere Availability Zone in der angeforderten Region verfügbar ist.

Beispiele

AWS CLI Beispiele für Befehle

Um einen AWS CLI Befehl idempotent zu machen, fügen Sie die `--client-token` Option hinzu.

Beispiel 1: Idempotenz

Der folgende Befehl [allocate-hosts](#) verwendet Idempotenz, da er ein Client-Token enthält.

```
aws ec2 allocate-hosts --instance-type m5.large --availability-zone eu-west-1a --auto-placement on --quantity 1 --client-token 550e8400-e29b-41d4-a716-446655440000
```

Beispiel 2: Regionale Idempotenz von Run-Instances

Der folgende Befehl [run-instances](#) verwendet regionale Idempotenz, da er ein Client-Token beinhaltet, aber weder explizit noch implizit eine Availability Zone spezifiziert.

```
aws ec2 run-instances --image-id ami-b232d0db --count 1 --key-name my-key-pair --client-token 550e8400-e29b-41d4-a716-446655440000
```

Beispiel 3: Zonale Idempotenz von Run-Instances

Der folgende Befehl [run-instances](#) verwendet zonale Idempotenz, da er ein Client-Token und eine explizit angegebene Availability Zone enthält.

```
aws ec2 run-instances --placement "AvailabilityZone=us-east-1a" --image-id ami-b232d0db --count 1 --key-name my-key-pair --client-token 550e8400-e29b-41d4-a716-446655440000
```

API Beispiele anfordern

Um eine API Anfrage idempotent zu machen, fügen Sie den ClientToken Parameter hinzu.

Beispiel 1: Idempotenz

Die folgende [AllocateHosts](#) API-Anfrage verwendet Idempotenz, da sie ein Client-Token enthält.

```
https://ec2.amazonaws.com/?Action=AllocateHosts  
&AvailabilityZone=us-east-1b  
&InstanceType=m5.large
```

```
&Quantity=1
&AutoPlacement=off
&ClientToken=550e8400-e29b-41d4-a716-446655440000
&AUTHPARAMS
```

Beispiel 2: regionale Idempotenz RunInstances

Die folgende [RunInstances](#)APIAnfrage verwendet regionale Idempotenz, da sie ein Client-Token enthält, aber weder explizit noch implizit eine Availability Zone spezifiziert.

```
https://ec2.amazonaws.com/?Action=RunInstances
&ImageId=ami-3ac33653
&MaxCount=1
&MinCount=1
&KeyName=my-key-pair
&ClientToken=550e8400-e29b-41d4-a716-446655440000
&AUTHPARAMS
```

Beispiel 3: Zonale Idempotenz RunInstances

Die folgende [RunInstances](#)APIAnfrage verwendet zonale Idempotenz, da sie ein Client-Token und eine explizit angegebene Availability Zone enthält.

```
https://ec2.amazonaws.com/?Action=RunInstances
&Placement.AvailabilityZone=us-east-1d
&ImageId=ami-3ac33653
&MaxCount=1
&MinCount=1
&KeyName=my-key-pair
&ClientToken=550e8400-e29b-41d4-a716-446655440000
&AUTHPARAMS
```

Versuchen Sie es erneut mit Empfehlungen für idempotente Anfragen

Die folgende Tabelle zeigt einige häufig auftretende Antworten auf idempotente API Anfragen und enthält Empfehlungen für Wiederholungen.

Antwort	Empfehlung	Kommentare
200 (OK)		

Antwort	Empfehlung	Kommentare
	Nicht erneut versuchen	Die ursprüngliche Anfrage wurde erfolgreich abgeschlossen. Alle nachfolgenden Wiederholungsversuche werden als erfolgreich zurückgegeben.
Antwortcodes der Serie 400 (Client-Fehler)	Nicht erneut versuchen	<p>Es liegt eins der folgenden Probleme mit der Anfrage vor:</p> <ul style="list-style-type: none"> • Sie enthält einen Parameter oder eine Parameterkombination, der/die nicht gültig ist. • Sie verwendet eine Aktion oder Ressource, für die Sie keine Berechtigungen haben. • Sie verwendet eine Ressource, deren Status sich gerade ändert. <p>Wenn die Anfrage eine Ressource umfasst, deren Status sich gerade ändert, könnte ein erneuter Anfrageversuch möglicherweise erfolgreich sein.</p>
Antwortcodes der Serie 500 (Serverfehler)	Erneut versuchen	Der Fehler wird durch ein AWS serverseitiges Problem verursacht und ist im Allgemeinen vorübergehend. Wiederholen Sie die Anfrage mit einer geeigneten Backoff-Strategie.

Drosselung für Amazon beantragen EC2 API

Amazon EC2 drosselt EC2 API Anfragen für jedes AWS Konto pro Region. Wir tun dies, um die Leistung des Dienstes zu verbessern und eine faire Nutzung für alle EC2 Amazon-Kunden

sicherzustellen. Durch die Drosselung wird sichergestellt, dass Anfragen an Amazon die maximal zulässigen API Anforderungslimits EC2 API nicht überschreiten. APIAnfragen unterliegen den Anforderungsbeschränkungen, unabhängig davon, ob sie von folgenden Quellen stammen:

- Eine Anwendung eines Drittanbieters
- Ein Befehlszeilentool
- Die EC2 Amazon-Konsole

Wenn Sie ein API Drosselungslimit überschreiten, erhalten Sie den `RequestLimitExceeded` Fehlercode.

Inhalt

- [Wie wird die Drosselung angewendet](#)
- [Drosselungsgrenzwerte](#)
- [Drosselung überwachen API](#)
- [Wiederholungen und exponentieller Backoff](#)
- [Anfordern einer -Limit-Erhöhung](#)

Wie wird die Drosselung angewendet

Amazon EC2 verwendet den [Token-Bucket-Algorithmus](#), um die API Drosselung zu implementieren. Mit diesem Algorithmus verfügt Ihr Konto über einen Bucket, der eine bestimmte Anzahl von Token enthält. Die Anzahl der Token im Bucket entspricht Ihrem Drosselungslimit für eine bestimmte Sekunde.

Amazon EC2 implementiert zwei Arten der API Drosselung:

APIArten der Drosselung

- [Anforderungsratenbegrenzung](#)
- [Begrenzung der Ressourcenrate](#)

Anforderungsratenbegrenzung

Bei der Begrenzung der Anforderungsrate API wird jede Anfrage einzeln bewertet, und die Anzahl der Anfragen, die Sie pro Anfrage stellen, wird eingeschränkt. API Jede Anfrage, die Sie stellen, entfernt ein Token aus dem s-BucketAPI. Beispielsweise beträgt die Token-Bucket-Größe für eine

API Aktion `DescribeHosts`, die nicht mutiert, 100 Token. Sie können in einer Sekunde bis zu 100 `DescribeHosts` Anfragen stellen. Wenn Sie in einer Sekunde mehr als 100 Anfragen haben, wird dieser Wert eingeschränkt API und die verbleibenden Anfragen innerhalb dieser Sekunde schlagen fehl. Anfragen für andere API sind davon jedoch nicht betroffen.

Eimer werden automatisch mit einer bestimmten Geschwindigkeit wieder aufgefüllt. Wenn der Bucket seine maximale Kapazität unterschreitet, wird ihm jede Sekunde eine bestimmte Anzahl von Tokens hinzugefügt, bis er seine maximale Kapazität erreicht hat. Wenn der Eimer voll ist, wenn die Nachfüll-Token eintreffen, werden sie weggeworfen. Der Eimer kann nicht mehr als die maximale Anzahl an Tokens aufnehmen. Beispielsweise beträgt die Bucket-Größe für eine API Aktion `DescribeHosts`, die nicht mutiert, 100 Token und die Nachfüllrate beträgt 20 Token pro Sekunde. Wenn Sie in einer Sekunde 100 `DescribeHosts` Anfragen stellen, wird der Bucket auf null (0) Token reduziert. Der Bucket wird dann jede Sekunde um 20 Token aufgefüllt, bis er seine maximale Kapazität von 100 Token erreicht hat. Das bedeutet, dass ein leerer Bucket nach 5 Sekunden seine maximale Kapazität erreicht, wenn während dieser Zeit keine Anfragen gestellt werden.

Sie müssen nicht warten, bis der Bucket vollständig gefüllt ist, bevor Sie API Anfragen stellen können. Sie können Nachfüll-Token verwenden, wenn sie dem Bucket hinzugefügt werden. Wenn Sie die Nachfüll-Token sofort verwenden, erreicht der Bucket nicht seine maximale Kapazität. Beispielsweise beträgt die Bucketgröße für eine API Aktion `DescribeHosts`, die nicht mutiert, 100 Tokens und die Nachfüllrate 20 Tokens pro Sekunde. Wenn Sie den Bucket leeren, indem Sie 100 API Anfragen pro Sekunde stellen, können Sie weiterhin 20 API Anfragen pro Sekunde stellen, indem Sie die Nachfüll-Token verwenden, wenn sie dem Bucket hinzugefügt werden. Der Bucket kann nur dann bis zur maximalen Kapazität aufgefüllt werden, wenn Sie weniger als 20 API Anfragen pro Sekunde stellen.

Begrenzung der Ressourcenrate

Bei einigen API Aktionen, wie z. B. `RunInstances` und `TerminateInstances`, wie in der folgenden Tabelle beschrieben, wird zusätzlich zur Begrenzung der Anforderungsrate eine Begrenzung der Ressourcenrate verwendet. Für diese API Aktionen gibt es einen separaten Ressourcentoken-Bucket, der je nach Anzahl der Ressourcen, die von der Anforderung betroffen sind, aufgebraucht wird. Wie bei Anforderungstoken-Buckets gibt es auch bei Ressourcentoken-Buckets eine maximale Anzahl an Buckets, sodass Sie überlastet werden können, und eine Nachfüllrate, die es Ihnen ermöglicht, eine konstante Anzahl von Anfragen so lange wie nötig aufrechtzuerhalten. Wenn Sie ein bestimmtes Bucket-Limit für einen überschreiten API, auch wenn ein Bucket noch nicht aufgefüllt wurde, um die nächste API Anfrage zu unterstützen, API ist die Aktion von eingeschränkt, obwohl Sie das Gesamtlimit für die Drosselung noch nicht erreicht haben.

API

Beispielsweise RunInstances beträgt die Bucket-Größe für Ressourcentokens 1000 Token, und die Nachfüllrate beträgt zwei Token pro Sekunde. Daher können Sie sofort 1000 Instances starten und dabei eine beliebige Anzahl von API Anfragen verwenden, z. B. eine Anfrage für 1000 Instances oder vier Anfragen für 250 Instances. Wenn der Ressourcentoken-Bucket leer ist, können Sie bis zu zwei Instances pro Sekunde starten, indem Sie entweder eine Anfrage für zwei Instances oder zwei Anfragen für eine Instance verwenden.

Weitere Informationen finden Sie unter [Größen und Nachfüllraten von Buckets für Ressourcentokens](#).

Drosselungsgrenzwerte

In den folgenden Abschnitten werden die Größen und Nachfüllraten der Buckets für Anforderungstoken und Ressourcentokens beschrieben.

Einschränkungen

- [Größen und Nachfüllraten für Token-Buckets anfordern](#)
- [Größen und Nachfüllraten von Buckets für Ressourcentokens](#)

Größen und Nachfüllraten für Token-Buckets anfordern

Zur Begrenzung der Anforderungsrate werden die API Aktionen in die folgenden Kategorien eingeteilt:

- Nicht mutierende Aktionen — API Aktionen, die Daten über Ressourcen abrufen. Diese Kategorie umfasst im Allgemeinen alle Describe* List*Search*, und Get* API Aktionen wie DescribeRouteTablesSearchTransitGatewayRoutes, und. GetIpamPoolCidrs Für diese API Aktionen gelten in der Regel die höchsten API Drosselungsgrenzen.
- Ungefilterte und nicht paginierte, nicht mutierende Aktionen — [Eine bestimmte Untergruppe nicht mutierender API Aktionen, die, wenn sie ohne Angabe einer Paginierung oder eines Filters angefordert werden, Token aus einem kleineren Token-Bucket verwenden](#). Es wird empfohlen, Seitennummerierung und Filterung zu verwenden, sodass Tokens vom standardmäßigen (größeren) Token-Bucket abgezogen werden.
- Mutierende Aktionen — API Aktionen, die Ressourcen erstellen, ändern oder löschen. Diese Kategorie umfasst im Allgemeinen alle API Aktionen, die nicht als nicht mutierende Aktionen eingestuft sind, wie z. B., AllocateHosts undModifyHosts. CreateCapacityReservation Diese Aktionen haben eine niedrigere Drosselungsgrenze als Aktionen, die nicht mutieren. API

- **Ressourcenintensive Aktionen** — Mutierte API Aktionen, deren Ausführung am meisten Zeit in Anspruch nimmt und die meisten Ressourcen verbraucht. Diese Aktionen haben eine noch niedrigere Drosselungsgrenze als mutierende Aktionen. Sie werden getrennt von anderen mutierenden Aktionen gedrosselt.
- **Nicht mutierende Konsolenaktionen** — Nicht mutierende API Aktionen, die von der Amazon-Konsole angefordert werden. EC2 Diese API Aktionen werden getrennt von anderen Aktionen, die nicht mutieren, gedrosselt. API
- **Unkategorisierte Aktionen** — Dabei handelt es sich um API Aktionen, die ihre eigenen Token-Bucketgrößen und Wiederauffüllraten erhalten, obwohl sie per Definition in eine der anderen Kategorien passen.

In der folgenden Tabelle sind die Bucketgrößen und Nachfüllraten für Anforderungstoken für alle AWS Regionen aufgeführt.

APIAktionenskategorie	Aktionen	Maximale Kapazität des Buckets	Nachfüllrate des Eimers
Aktionen, die nicht mutieren	<i>AlleDescribe*</i> , „, und <i>Get*</i> API Aktionen <i>List*Search*</i> , bei denen es sich nicht um ungefilterte und nicht paginierte, nicht mutierende Aktionen oder Unkategorisierte Aktionen handelt.	100	20
Ungefilterte und nicht paginierte, nicht mutierende Aktionen	<ul style="list-style-type: none"> • <code>DescribeInstances</code> • <code>DescribeInstanceStatus</code> 	50	10

APIAktionenkategorie	Aktionen	Maximale Kapazität des Buckets	Nachfüllrate des Eimers
	<ul style="list-style-type: none"> • DescribeNetworkInterfaces • DescribeSecurityGroups • DescribeSnapshots • DescribeSpotInstanceRequests • DescribeVolumes 		
Mutierende Aktionen	Alle mutierenden API Aktionen, bei denen es sich nicht um ressourcenintensive Aktionen oder um nicht kategorisierte Aktionen handelt.	50	5

APIAktionenskategorie	Aktionen	Maximale Kapazität des Buckets	Nachfüllrate des Eimers
Ressourcenintensive Aktionen	<ul style="list-style-type: none">• AcceptVpcPeeringConnection• AuthorizeSecurityGroupIngress• CancelSpotInstanceRequests• CreateKeyPair• CreateVpcPeeringConnection• DeleteVpcPeeringConnection• RejectVpcPeeringConnection• RevokeSecurityGroupIngress• RequestSpotInstances	50	5

APIAktionenskategorie	Aktionen	Maximale Kapazität des Buckets	Nachfüllrate des Eimers
Aktionen auf der Konsole, die nicht mutieren	<ul style="list-style-type: none"> Describe* Get* 	100	10
Nicht kategorisierte Aktionen	AcceptVpcEndpointConnections	10	1
	AdvertiseByoipCidr	1	0.1
	AssignIpv6Addresses	100	5
	AssignPrivateIpAddresses	100	5
	AssignPrivateNatGatewayAddress	10	1
	AssociateEnclaveCertificateIamRole	10	1
	AssociateIamInstanceProfile	100	5
	AssociateNatGatewayAddress	10	1

APIAktionskategorie	Aktionen	Maximale Kapazität des Buckets	Nachfüllrate des Eimers
	AttachVerifiedAccessTrustProvider	10	2
	CreateDefaultSubnet	1	1
	CreateDefaultVpc	1	1
	CopyImage	100	1
	CreateLaunchTemplateVersion	100	5
	CreateNatGateway	10	1
	CreateNetworkInterface	100	5
	CreateRestoreImageTask	50	0.1
	CreateSnapshot	100	5
	CreateSnapshots	100	5
	CreateStoreImageTask	50	0.1
	CreateTags	100	10

APIAktionenskategorie	Aktionen	Maximale Kapazität des Buckets	Nachfüllrate des Eimers
	CreateVerifiedAccessEndpoint	20	4
	CreateVerifiedAccessGroup	10	2
	CreateVerifiedAccessInstance	10	2
	CreateVerifiedAccessTrustProvider	10	2
	CreateVolume	100	5
	CreateVpcEndpoint	4	0.3
	CreateVpcEndpointServiceConfiguration	10	1
	DeleteNatGateway	10	1
	DeleteNetworkInterface	100	5
	DeleteSnapshot	100	5
DeleteTags	100	10	

APIAktionskategorie	Aktionen	Maximale Kapazität des Buckets	Nachfüllrate des Eimers
	DeleteQueuedReservedInstances	5	5
	DeleteVerifiedAccessEndpoint	20	4
	DeleteVerifiedAccessGroup	10	2
	DeleteVerifiedAccessInstance	10	2
	DeleteVerifiedAccessTrustProvider	10	2
	DeleteVolume	100	5
	DeleteVpcEndpoints	4	0.3
	DeleteVpcEndpointServiceConfigurations	10	1
	DeprovisionByoipCidr	1	0.1
	DeregisterImage	100	5

APIAktionskategorie	Aktionen	Maximale Kapazität des Buckets	Nachfüllrate des Eimers
	DetachVerifiedAccessTrustProvider	10	2
	DescribeByoipCidrs	1	0.5
	DescribeCapacityBlockOfferings	10	0.15
	DescribeInstanceTopology	1	1
	DescribeMovingAddresses	1	1
	DescribeReservedInstancesOfferings	10	10
	DescribeSpotFleetRequestHistory	100	5
	DescribeSpotFleetInstances	100	5
	DescribeSpotFleetRequests	50	3

APIAktionenskategorie	Aktionen	Maximale Kapazität des Buckets	Nachfüllrate des Eimers
	DescribeStoreImageTasks	50	0.5
	DescribeVerifiedAccessInstanceLoggingConfigurations	10	2
	DisableFastLaunch	5	2
	DisableImageBlockPublicAccess	1	0,1
	DisableSnapshotBlockPublicAccess	1	0.1
	DisassociateEnclaveCertificateIamRole	10	1
	DisassociateIamInstanceProfile	100	5
	DisassociateNatGatewayAddress	10	1

APIAktionskategorie	Aktionen	Maximale Kapazität des Buckets	Nachfüllrate des Eimers
	EnableFastLaunch	5	2
	EnableImageBlockPublicAccess	1	0,1
	EnableSnapshotBlockPublicAccess	1	0.1
	GetAssociatedEnclaveCertificateIamRoles	10	1
	ModifyImageAttribute	100	5
	ModifyInstanceMetadataOptions	100	5
	ModifyLaunchTemplate	100	5
	ModifyNetworkInterfaceAttribute	100	5
	ModifySnapshotAttribute	100	5
	ModifyVerifiedAccessEndpoint	20	4

APIAktionenskategorie	Aktionen	Maximale Kapazität des Buckets	Nachfüllrate des Eimers
	ModifyVerifiedAccessEndpointPolicy	20	4
	ModifyVerifiedAccessGroup	10	2
	ModifyVerifiedAccessGroupPolicy	20	4
	ModifyVerifiedAccessInstance	10	2
	ModifyVerifiedAccessInstanceLoggingConfiguration	10	2
	ModifyVerifiedAccessTrustProvider	10	2
	ModifyVpcEndpoint	4	0.3
	ModifyVpcEndpointServiceConfiguration	10	1

APIAktionenskategorie	Aktionen	Maximale Kapazität des Buckets	Nachfüllrate des Eimers
	MoveAddressToVpc	1	1
	ProvisionByoipCidr	1	0.1
	PurchaseCapacityBlock	10	0,15
	PurchaseReservedInstancesOffering	5	5
	RejectVpcEndpointConnections	10	1
	RestoreAddressToClassic	1	1
	RunInstances	5	2
	StartInstances	5	2
	TerminateInstances	100	5
	UnassignPrivateIpAddresses	100	5
	UnassignPrivateNatGatewayAddress	10	1

APIAktionenskategorie	Aktionen	Maximale Kapazität des Buckets	Nachfüllrate des Eimers
	WithdrawByoipCidr	1	0.1

Größen und Nachfüllraten von Buckets für Ressourcentokens

In der folgenden Tabelle sind die Bucket-Größen und Nachfüllraten für Ressourcentokens für API Aktionen aufgeführt, bei denen die Begrenzung der Ressourcenrate verwendet wird.

APIAktion	Maximale Kapazität des Buckets	Nachfüllrate des Eimers
RunInstances	1000	2
TerminateInstances	1000	20
StartInstances	1000	2
StopInstances	1000	20

Drosselung überwachen API

Sie können Amazon verwenden CloudWatch , um Ihre EC2 API Amazon-Anfragen zu überwachen und Messwerte rund um API Drosselung zu sammeln und zu verfolgen. Sie können auch einen Alarm einrichten, der Sie warnt, wenn Sie kurz davor sind, die API Drosselungsgrenzen zu erreichen. Weitere Informationen finden Sie unter [Überwachen Sie EC2 API Amazon-Anfragen mit Amazon CloudWatch](#).

Wiederholungen und exponentieller Backoff

Ihre Anwendung muss möglicherweise eine Anfrage erneut versuchen. API Beispielsweise:

- Um zu überprüfen, ob der Status einer Ressource aktualisiert wurde
- Um eine große Anzahl von Ressourcen aufzuzählen (z. B. alle Ihre Volumes)

- Um eine Anfrage erneut zu versuchen, nachdem sie mit einem Serverfehler (5xx) oder einem Drosselungsfehler fehlschlägt

Bei einem Client-Fehler (4xx) müssen Sie die Anfrage jedoch überarbeiten, um das Problem zu beheben, bevor Sie die Anfrage erneut versuchen.

Der Ressourcenstatus ändert sich

Bevor Sie mit der Abfrage beginnen, um nach Statusaktualisierungen zu suchen, sollten Sie der Anfrage Zeit geben, bis sie möglicherweise abgeschlossen ist. Warten Sie beispielsweise einige Minuten, bevor Sie überprüfen, ob Ihre Instance aktiv ist. Wenn Sie mit dem Polling beginnen, sollten Sie ein angemessenes Schlafintervall zwischen aufeinanderfolgenden Anfragen verwenden, um die Anzahl der API Anfragen zu verringern. Um die besten Ergebnisse zu erzielen, verwenden Sie ein zunehmendes oder variables Energiesparintervall.

Alternativ können Sie Amazon verwenden, EventBridge um Sie über den Status einiger Ressourcen zu informieren. Sie können beispielsweise das Ereignis EC2Instance State-Change Notification verwenden, um Sie über eine Statusänderung einer Instance zu informieren. Weitere Informationen finden Sie unter [Automatisieren EC2 von Amazon mithilfe EventBridge](#).

Wiederholversuche

Wenn Sie eine API Anfrage abfragen oder erneut versuchen müssen, empfehlen wir die Verwendung eines exponentiellen Backoff-Algorithmus zur Berechnung des Schlafintervalls zwischen Anfragen. Die Idee hinter dem exponentiellen Backoff ist, bei aufeinanderfolgenden Fehlermeldungen progressiv längere Wartezeiten zwischen den Wiederholversuchen zu verwenden. Sie sollten ein maximales Verzögerungsintervall sowie eine maximale Anzahl von Wiederholversuchen implementieren. Sie können auch Jitter (randomisierte Verzögerung) verwenden, um aufeinanderfolgende Kollisionen zu verhindern. Weitere Informationen finden Sie unter [Timeouts, Wiederholungen](#) und Backoff mit Jitter.

In beiden Fällen wird eine automatische Wiederholungslogik implementiert AWS SDK. Weitere Informationen finden Sie unter [Verhalten bei Wiederholungsversuchen](#) im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

Anfordern einer -Limit-Erhöhung

Sie können eine Erhöhung der API Drosselungsgrenzen für Ihren beantragen. AWS-Konto

Um Zugriff auf diese Funktion zu beantragen

1. [AWS Support Zentrum](#) öffnen.
2. Wählen Sie Create case (Fall erstellen) aus.
3. Wählen Sie Konto und Fakturierung aus.
4. Wählen Sie für Service die Optionen Allgemeine Informationen und Erste Schritte aus.
5. Wählen Sie als Kategorie die Option Nutzung AWS und Dienste aus.
6. Wählen Sie Next step: Additional information (Nächster Schritt: Zusätzliche Informationen).
7. Geben Sie unter Subject (Betreff) **Request an increase in my Amazon EC2 API throttling limits** ein.
8. Geben Sie für Beschreibung den Text **Please increase the API throttling limits for my account. Related page: <https://docs.aws.amazon.com/AWSEC2/latest/APIReference/throttling.html>** ein. Schließen Sie außerdem die folgenden Informationen ein:
 - Eine Beschreibung Ihres Anwendungsfalls
 - Die Regionen, in denen Sie eine Erhöhung benötigen.
 - Das einstündige ZeitfensterUTC, in dem die Drosselung oder Auslastung zu Spitzenzeiten auftrat (zur Berechnung der neuen Drosselungsgrenze).
9. Klicken Sie auf Next step: Solve now or contact us () (Nächster Schritt): Jetzt lösen oder Support kontaktieren).
10. Wählen Sie auf der Registerkarte Kontakt Ihre bevorzugte Kontaktsprache und Kontaktmethode aus.
11. Wählen Sie Absenden aus.

Erstellen Sie Amazon EC2 EC2-Ressourcen mit dem AWS CLI

Sie können Ihre Amazon EC2 EC2-Ressourcen mithilfe von AWS Command Line Interface (AWS CLI) in einer Befehlszeilen-Shell erstellen und verwalten. Das AWS CLI bietet direkten Zugriff auf die APIs für AWS-Services, z. B. Amazon EC2.

Syntax und Beispiele für die Befehle für Amazon EC2 finden Sie unter [ec2](#) in der AWS CLI Befehlsreferenz. Sie finden diese Beispiele auch in [aws-cli/awscli/examples/ec2](#) auf Github.

Erfahren Sie mehr über AWS CLI

Weitere Informationen über die AWS CLI finden Sie in den folgenden Ressourcen:

- [AWS Command Line Interface](#)
- [AWS Command Line Interface Benutzerhandbuch für Version 2](#)
- [AWS Command Line Interface Benutzerhandbuch für Version 1](#)

Erstellen Sie Amazon EC2 EC2-Ressourcen mit AWS CloudFormation

Amazon EC2 ist integriert mit AWS CloudFormation, ein Service, der Sie bei der Modellierung und Einrichtung Ihrer AWS Ressourcen unterstützt, sodass Sie weniger Zeit mit der Erstellung und Verwaltung Ihrer Ressourcen und Infrastruktur verbringen müssen. Sie erstellen eine Vorlage, die die benötigten AWS Ressourcen (wie Instances und Subnetze) beschreibt und diese Ressourcen für Sie mit AWS CloudFormation bereitstellt und konfiguriert.

Wenn Sie sie verwenden AWS CloudFormation, können Sie Ihre Vorlage wiederverwenden, um Ihre Amazon EC2 EC2-Ressourcen konsistent und wiederholt einzurichten. Beschreiben Sie Ihre Ressourcen einmal und stellen Sie dann dieselben Ressourcen immer wieder in mehreren Regionen AWS-Konten bereit.

Amazon EC2 und Vorlagen AWS CloudFormation

Um Ressourcen für Amazon EC2 und verwandte Services bereitzustellen und zu konfigurieren, müssen Sie [AWS CloudFormation Vorlagen](#) verstehen. Vorlagen sind formatierte Textdateien in JSON oder YAML. Diese Vorlagen beschreiben die Ressourcen, die Sie in Ihren AWS CloudFormation Stacks bereitstellen werden. Wenn Sie mit JSON oder YAML nicht vertraut sind, können Sie AWS CloudFormation Designer verwenden, um Ihnen bei den ersten Schritten mit Vorlagen zu helfen. AWS CloudFormation Weitere Informationen finden Sie unter [Was ist AWS CloudFormation Designer?](#) im AWS CloudFormation Benutzerhandbuch.

Ressourcen für Amazon EC2

Ressourcen berechnen

- [AWS::EC2::CapacityReservation](#)
- [AWS::EC2::CapacityReservationFlotte](#)
- [AWS::EC2::EC2Fleet](#)
- [AWS::EC2::EC2Fleet](#)
- [AWS::EC2::Host](#)
- [AWS::EC2::Instance](#)
- [AWS::EC2::InstanceConnectEndpoint](#)

- [AWS::EC2::LaunchTemplate](#)
- [AWS::EC2::PlacementGroup](#)
- [AWS::EC2::SpotFleet](#)

Netzwerkressourcen

- [AWS::EC2::CarrierGateway](#)
- [AWS::EC2::ClientVpnAuthorizationRule](#)
- [AWS::EC2::ClientVpnEndpoint](#)
- [AWS::EC2::ClientVpnRoute](#)
- [AWS::EC2::ClientVpnTargetNetworkAssociation](#)
- [AWS::EC2::CustomerGateway](#)
- [AWS::EC2::DHCPOptions](#)
- [AWS::EC2::EgressOnlyInternetGateway](#)
- [AWS::EC2::EIP](#)
- [AWS::EC2::EIPAssociation](#)
- [AWS::EC2::FlowLog](#)
- [AWS::EC2::GatewayRouteTableAssociation](#)
- [AWS::EC2::InternetGateway](#)
- [AWS::EC2::IPAM](#)
- [AWS::EC2::IPAMAllocation](#)
- [AWS::EC2::IPAMPool](#)
- [AWS::EC2::IPAMPoolCidr](#)
- [AWS::EC2::IPAMResourceDiscovery](#)
- [AWS::EC2::IPAMResourceDiscoveryAssociation](#)
- [AWS::EC2::IPAMScope](#)
- [AWS::EC2::LocalGatewayRoute](#)
- [AWS::EC2::LocalGatewayRouteTable](#)
- [AWS::EC2::LocalGatewayRouteTableVirtualInterfaceGroupAssociation](#)
- [AWS::EC2::LocalGatewayRouteTableVPC-Verband](#)
- [AWS::EC2::NatGateway](#)

- [AWS::EC2::NetworkInterface](#)
- [AWS::EC2::NetworkInsightsAccessScope](#)
- [AWS::EC2::NetworkInsightsAccessScopeAnalysis](#)
- [AWS::EC2::NetworkInsightsAnalyse](#)
- [AWS::EC2::NetworkInsightsPfad](#)
- [AWS::EC2::NetworkInterfaceAnlage](#)
- [AWS::EC2::NetworkInterfaceErlaubnis](#)
- [AWS::EC2::NetworkPerformanceMetricSubscription](#)
- [AWS::EC2::PrefixList](#)
- [AWS::EC2::Route](#)
- [AWS::EC2::RouteTable](#)
- [AWS::EC2::Subnet](#)
- [AWS::EC2::SubnetCidrBlockieren](#)
- [AWS::EC2::SubnetNetworkAclAssociation](#)
- [AWS::EC2::SubnetRouteTableAssociation](#)
- [AWS::EC2::TrafficMirrorFiltern](#)
- [AWS::EC2::TrafficMirrorFilterRule](#)
- [AWS::EC2::TrafficMirrorSitzung](#)
- [AWS::EC2::TrafficMirrorZiel](#)
- [AWS::EC2::TransitGateway](#)
- [AWS::EC2::TransitGatewayAnlage](#)
- [AWS::EC2::TransitGatewayConnect](#)
- [AWS::EC2::TransitGatewayMulticastDomain](#)
- [AWS::EC2::TransitGatewayMulticastDomainAssociation](#)
- [AWS::EC2::TransitGatewayMulticastGroupMember](#)
- [AWS::EC2::TransitGatewayMulticastGroupSource](#)
- [AWS::EC2::TransitGatewayPeeringAttachment](#)
- [AWS::EC2::TransitGatewayRoute](#)
- [AWS::EC2::TransitGatewayRouteTable](#)
- [AWS::EC2::TransitGatewayRouteTableAssociation](#)

- [AWS::EC2::TransitGatewayRouteTablePropagation](#)
- [AWS::EC2::TransitGatewayVpcAttachment](#)
- [AWS::EC2::VPC](#)
- [AWS::EC2::VPCcidrBlock](#)
- [AWS::EC2::VPCDHCP OptionsAssociation](#)
- [AWS::EC2::VPCEndpoint](#)
- [AWS::EC2::VPCEndpointConnectionNotification](#)
- [AWS::EC2::VPCEndpointService](#)
- [AWS::EC2::VPCEndpointServicePermissions](#)
- [AWS::EC2::VPCGatewayAttachment](#)
- [AWS::EC2::VPCPeeringConnection](#)
- [AWS::EC2::VPNConnection](#)
- [AWS::EC2::VPN ConnectionRoute](#)
- [AWS::EC2::VPNGateway](#)
- [AWS::EC2::VPN GatewayRoutePropagation](#)

Ressourcen zum Thema Sicherheit

- [AWS::EC2::KeyPair](#)
- [AWS::EC2::NetworkAcl](#)
- [AWS::EC2::NetworkAclEintrag](#)
- [AWS::EC2::SecurityGroup](#)
- [AWS::EC2::SecurityGroupAustritt](#)
- [AWS::EC2::SecurityGroupEintritt](#)
- [AWS::EC2::VerifiedAccessEndpunkt](#)
- [AWS::EC2::VerifiedAccessGruppe](#)
- [AWS::EC2::VerifiedAccessInstanz](#)
- [AWS::EC2::VerifiedAccessTrustProvider](#)

Speicherressourcen

- [AWS::EC2::SnapshotBlockPublicAccess](#)

- [AWS::EC2::Volume](#)
- [AWS::EC2::VolumeAttachment](#)

Erfahre mehr über AWS CloudFormation

Weitere Informationen AWS CloudFormation dazu finden Sie in den folgenden Ressourcen:

- [AWS CloudFormation](#)
- [AWS CloudFormation Benutzerhandbuch](#)

Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK

AWS stellt Softwareentwicklungskits (SDK) für viele beliebte Programmiersprachen bereit. An SDK macht die Entwicklung effizienter, indem es Folgendes bietet:

- Vorgefertigte Komponenten und Bibliotheken, die Sie in Ihre Anwendungen integrieren können
- Sprachspezifische Tools wie Compiler und Debugger
- Kryptografisches Signieren von Serviceanfragen
- Wiederholungsversuche anfordern
- Behandlung von Fehlern und Antworten

Codebeispiele für den Amazon EC2 API

Die von bereitgestellten Codebeispiele AWS zeigen Ihnen, wie Sie einen verwenden API und bestimmte Aufgaben ausführen. Beispiele für Amazon finden Sie EC2 API unter [Codebeispiele für Amazon EC2](#). Weitere Beispiele [finden Sie unter Suchen Sie nach Codebeispielen für AWS SDKs oder aws-doc-sdk-exampleshttps://github.com/awsdocs/aws-doc-sdk-examples/](#) auf Github.

Erfahren Sie mehr über die AWS SDKs

Weitere Informationen über die AWS SDKs finden Sie in den folgenden Ressourcen:

- [AWS SDKsund Referenzhandbuch für Tools](#)
- [Tools, auf denen Sie aufbauen können AWS](#)
- [Was ist einSDK?](#)

Low-Level-API für Amazon EC2

Die Low-Level-API für Amazon EC2 ist die Schnittstelle auf Protokollebene für Amazon EC2. Wenn Sie die Low-Level-API verwenden, müssen Sie jede HTTPS-Anfrage korrekt formatieren und jeder Anfrage eine gültige digitale Signatur hinzufügen. Weitere Informationen finden Sie unter [Anfragen an die Amazon EC2 EC2-API stellen](#) in der Amazon EC2 EC2-API-Referenz. Alternativ können Sie ein AWS SDK verwenden, das die Anfragen in Ihrem Namen erstellt und signiert. Weitere Informationen finden Sie unter [Mit einem AWS SDK](#).

Die Amazon EC2 EC2-API besteht aus Aktionen und Datentypen für mehrere Services. Die Aktionen für jeden Service finden Sie auf den folgenden Seiten in der Amazon EC2 API-Referenz.

- [AWS Client VPN Aktionen](#)
- [Amazon EBS-Aktionen](#)
- [Amazon EC2 EC2-Aktionen](#)
- [AWS Network Manager Aktionen](#)
- [AWS Aktionen von Nitro Enclaves](#)
- [AWS Outposts Aktionen](#)
- [AWS PrivateLink Aktionen](#)
- [Aktionen im Papierkorb](#)
- [AWS Site-to-Site VPN Aktionen](#)
- [AWS Transit Gateway Aktionen](#)
- [AWS Verified Access Aktionen](#)
- [VM-Import-/Export-Aktionen](#)
- [Amazon VPC-Aktionen](#)
- [Amazon VPC IPAM-Aktionen](#)
- [AWS Wavelength Aktionen](#)

Verwenden Sie Console-to-Code, um Code für Ihre EC2 Konsolenaktionen zu generieren

Die Konsole bietet eine Anleitung zum Erstellen von Ressourcen und zum Testen von Prototypen. Wenn Sie dieselben Ressourcen in großem Umfang erstellen möchten, benötigen Sie Automatisierungscode. Console-to-Code ist eine Funktion von Amazon Q Developer, die Ihnen den Einstieg in Ihren Automatisierungscode erleichtern kann. Console-to-Code zeichnet Ihre Konsolenaktionen auf, einschließlich Standardeinstellungen und kompatibler Parameter. Anschließend schlägt es mithilfe generativer KI Code in Ihrem bevorzugten Format infrastructure-as-code (IaC) für die gewünschten Aktionen vor. Da der Konsolen-Workflow sicherstellt, dass die von Ihnen angegebenen Parameterwerte zusammen gültig sind, hat der Code, den Sie mithilfe von diesem Code generieren, kompatible Parameterwerte. Sie können den Code als Ausgangspunkt verwenden und ihn dann so anpassen, dass er für Ihren speziellen Anwendungsfall produktionsbereit ist.

Mit können Sie Console-to-Code beispielsweise den Start einer EC2 Amazon-Instance aufzeichnen und wählen, ob Code im AWS CloudFormation JSON Format generiert werden soll. Anschließend können Sie diesen Code kopieren und für die Verwendung in Ihrer AWS CloudFormation Vorlage anpassen.

Console-to-Code kann derzeit infrastructure-as-code (IaC) in den folgenden Sprachen und Formaten generieren:

- CDKJava
- CDKPython
- CDK TypeScript
- CloudFormation JSON
- CloudFormation YAML

Weitere Informationen und Anweisungen zur Verwendung Console-to-Code finden Sie unter [Automatisieren von AWS Services mit Amazon Q Developer Console-to-Code](#) im Amazon Q Developer User Guide.

Codebeispiele für Amazon EC2 mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Amazon EC2 mit einem AWS Software Development Kit (SDK) verwendet wird.

Basics sind Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Operationen innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Aktionen zeigen Ihnen zwar, wie Sie einzelne Servicefunktionen aufrufen, aber Sie können Aktionen im Kontext der zugehörigen Szenarien sehen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Dienstes oder in Kombination mit anderen aufrufen AWS-Services.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Erste Schritte

Hallo Amazon EC2

Die folgenden Codebeispiele zeigen, wie Sie mit Amazon beginnen können EC2.

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
namespace EC2Actions;  
  
public class HelloEc2  
{
```

```
/// <summary>
/// HelloEc2 lists the existing security groups for the default users.
/// </summary>
/// <param name="args">Command line arguments</param>
/// <returns>Async task.</returns>
static async Task Main(string[] args)
{
    // Set up dependency injection for Amazon Elastic Compute Cloud (Amazon
    EC2).
    using var host =
    Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
        .ConfigureServices( (_, services) =>
            services.AddAWSService<IAmazonEC2>()
                .AddTransient<EC2Wrapper>()
        )
        .Build();

    // Now the client is available for injection.
    var ec2Client = host.Services.GetRequiredService<IAmazonEC2>();

    try
    {
        // Retrieve information for up to 10 Amazon EC2 security groups.
        var request = new DescribeSecurityGroupsRequest { MaxResults = 10, };
        var securityGroups = new List<SecurityGroup>();

        var paginatorForSecurityGroups =
            ec2Client.Paginators.DescribeSecurityGroups(request);

        await foreach (var securityGroup in
            paginatorForSecurityGroups.SecurityGroups)
        {
            securityGroups.Add(securityGroup);
        }

        // Now print the security groups returned by the call to
        // DescribeSecurityGroupsAsync.
        Console.WriteLine("Welcome to the EC2 Hello Service example. " +
            "\nLet's list your Security Groups:");
        securityGroups.ForEach(group =>
        {
            Console.WriteLine(
                $"Security group: {group.GroupName} ID: {group.GroupId}");
        });
    }
}
```

```
    }
    catch (AmazonEC2Exception ex)
    {
        Console.WriteLine($"An Amazon EC2 service error occurred while
listing security groups. {ex.Message}");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"An error occurred while listing security groups.
{ex.Message}");
    }
}
}
```

- API-Einheiten finden Sie [DescribeSecurityGroups](#) unter AWS SDK for .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Code für die CMakeLists CMake .txt-Datei.

```
# 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 ec2)

# Set this project's name.
project("hello_ec2")

# 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_ec2.cpp)

target_link_libraries(${PROJECT_NAME}
  ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei hello_ec2.cpp.

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeInstancesRequest.h>
#include <iomanip>
#include <iostream>

/*
```

```
* A "Hello EC2" starter application which initializes an Amazon Elastic Compute
Cloud (Amazon EC2) client and describes
* the Amazon EC2 instances.
*
* main function
*
* Usage: 'hello_ec2'
*
*/

int main(int argc, char **argv) {
    (void)argc;
    (void)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";

        Aws::EC2::EC2Client ec2Client(clientConfig);
        Aws::EC2::Model::DescribeInstancesRequest request;
        bool header = false;
        bool done = false;
        while (!done) {
            Aws::EC2::Model::DescribeInstancesOutcome outcome =
ec2Client.DescribeInstances(request);
            if (outcome.IsSuccess()) {
                if (!header) {
                    std::cout << std::left <<
                        std::setw(48) << "Name" <<
                        std::setw(20) << "ID" <<
                        std::setw(25) << "Ami" <<
                        std::setw(15) << "Type" <<
                        std::setw(15) << "State" <<
                        std::setw(15) << "Monitoring" << std::endl;
                    header = true;
                }
            }

            const std::vector<Aws::EC2::Model::Reservation> &reservations =
```

```

        outcome.GetResult().GetReservations();

        for (const auto &reservation: reservations) {
            const std::vector<Aws::EC2::Model::Instance> &instances =
                reservation.GetInstances();
            for (const auto &instance: instances) {
                Aws::String instanceStateString =

Aws::EC2::Model::InstanceStateNameMapper::GetNameForInstanceStateName(
                    instance.GetState().GetName());

                Aws::String typeString =

Aws::EC2::Model::InstanceTypeMapper::GetNameForInstanceType(
                    instance.GetInstanceType());

                Aws::String monitorString =

Aws::EC2::Model::MonitoringStateMapper::GetNameForMonitoringState(
                    instance.GetMonitoring().GetState());
                Aws::String name = "Unknown";

                const std::vector<Aws::EC2::Model::Tag> &tags =
instance.GetTags();
                auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
                    [](const
Aws::EC2::Model::Tag &tag) {
                    return tag.GetKey() ==
"Name";
                });
                if (nameIter != tags.cend()) {
                    name = nameIter->GetValue();
                }
                std::cout <<
                    std::setw(48) << name <<
                    std::setw(20) << instance.GetInstanceId() <<
                    std::setw(25) << instance.GetImageId() <<
                    std::setw(15) << typeString <<
                    std::setw(15) << instanceStateString <<
                    std::setw(15) << monitorString << std::endl;
            }
        }

        if (!outcome.GetResult().GetNextToken().empty()) {

```



```

        request.SetNextToken(outcome.GetResult().GetNextToken());
    } else {
        done = true;
    }
} else {
    std::cerr << "Failed to describe EC2 instances:" <<
        outcome.GetError().GetMessage() << std::endl;
    result = 1;
    break;
}
}
}

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

```

- API-Einheiten finden Sie unter [DescribeSecurityGroups AWS SDK for C++ API Referenz](#).

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

/**
 * Asynchronously describes the security groups for the specified group ID.
 *
 * @param groupName the name of the security group to describe
 * @return a {@link CompletableFuture} that represents the asynchronous
operation
 *
 * of describing the security groups. The future will complete with a
 * {@link DescribeSecurityGroupsResponse} object that contains the
 * security group information.
 */

```

```
public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
    DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
        .groupNames(groupName)
        .build();

    DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
    AtomicReference<String> groupIdRef = new AtomicReference<>();
    return paginator.subscribe(response -> {
        response.securityGroups().stream()
            .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
            .findFirst()
            .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
    }).thenApply(v -> {
        String groupId = groupIdRef.get();
        if (groupId == null) {
            throw new RuntimeException("No security group found with the
name: " + groupName);
        }
        return groupId;
    }).exceptionally(ex -> {
        logger.info("Failed to describe security group: " + ex.getMessage());
        throw new RuntimeException("Failed to describe security group", ex);
    });
}
```

- API Einzelheiten finden Sie [DescribeSecurityGroups](#) unter AWS SDK for Java 2.x API Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import { DescribeSecurityGroupsCommand, EC2Client } from "@aws-sdk/client-ec2";

// Call DescribeSecurityGroups and display the result.
export const main = async () => {
  const client = new EC2Client();
  try {
    const { SecurityGroups } = await client.send(
      new DescribeSecurityGroupsCommand({}),
    );

    const securityGroupList = SecurityGroups.slice(0, 9)
      .map((sg) => ` • ${sg.GroupId}: ${sg.GroupName}`)
      .join("\n");

    console.log(
      "Hello, Amazon EC2! Let's list up to 10 of your security groups:",
    );
    console.log(securityGroupList);
  } catch (err) {
    console.error(err);
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- API-Einheiten finden Sie [DescribeSecurityGroups](#) unter AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->

        val response = ec2.describeSecurityGroups(request)
        response.securityGroups?.forEach { group ->
            println("Found Security Group with id ${group.groupId}, vpc id
                ${group.vpcId} and description ${group.description}")
        }
    }
}
```

- API-Einheiten finden Sie [DescribeSecurityGroups](#) in der AWS SDK API Kotlin-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
def hello_ec2(ec2_client):
    """
    Use the AWS SDK for Python (Boto3) to list the security groups in your
    account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ec2_client: A Boto3 EC2 client. This client provides low-level
        access to AWS EC2 services.
    """
    print("Hello, Amazon EC2! Let's list up to 10 of your security groups:")
    try:
        paginator = ec2_client.get_paginator("describe_security_groups")
        response_iterator = paginator.paginate(MaxResults=10)
        for page in response_iterator:
            for sg in page["SecurityGroups"]:
                logger.info(f"\t{sg['GroupId']}: {sg['GroupName']}")
    except ClientError as err:
        logger.error("Failed to list security groups.")
        if err.response["Error"]["Code"] == "AccessDeniedException":
            logger.error("You do not have permission to list security groups.")
        raise

if __name__ == "__main__":
    hello_ec2(boto3.client("ec2"))
```

- API Einzelheiten finden Sie unter [DescribeSecurityGroups Python \(Boto3\) API - Referenz](#). AWS SDK

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
require 'aws-sdk-ec2'
require 'logger'

# EC2Manager is a class responsible for managing EC2 operations
# such as listing all EC2 instances in the current AWS account.
class EC2Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all EC2 instances in the current AWS account.
  def list_instances
    @logger.info('Listing instances')

    instances = fetch_instances

    if instances.empty?
      @logger.info('You have no instances')
    else
      print_instances(instances)
    end
  end

  private

  # Fetches all EC2 instances using pagination.
  #
  # @return [Array<Aws::EC2::Types::Instance>] List of EC2 instances.
  def fetch_instances
    paginator = @client.describe_instances
```

```
instances = []

paginator.each_page do |page|
  page.reservations.each do |reservation|
    reservation.instances.each do |instance|
      instances << instance
    end
  end
end

instances

end

# Prints details of the given EC2 instances.
#
# @param instances [Array<Aws::EC2::Types::Instance>] List of EC2 instances to
print.
def print_instances(instances)
  instances.each do |instance|
    @logger.info("Instance ID: #{instance.instance_id}")
    @logger.info("Instance Type: #{instance.instance_type}")
    @logger.info("Public IP: #{instance.public_ip_address}")
    @logger.info("Public DNS Name: #{instance.public_dns_name}")
    @logger.info("\n")
  end
end
end

if $PROGRAM_NAME == __FILE__
  ec2_client = Aws::EC2::Client.new(region: 'us-west-2')
  manager = EC2Manager.new(ec2_client)
  manager.list_instances
end
```

- API Einheiten finden Sie [DescribeSecurityGroups](#) unter AWS SDK for Ruby API Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
async fn show_security_groups(client: &aws_sdk_ec2::Client, group_ids:
Vec<String>) {
    let response = client
        .describe_security_groups()
        .set_group_ids(Some(group_ids))
        .send()
        .await;

    match response {
        Ok(output) => {
            for group in output.security_groups() {
                println!(
                    "Found Security Group {} ({}), vpc id {} and description {}",
                    group.group_name().unwrap_or("unknown"),
                    group.group_id().unwrap_or("id-unknown"),
                    group.vpc_id().unwrap_or("vpcid-unknown"),
                    group.description().unwrap_or("(none)")
                );
            }
        }
        Err(err) => {
            let err = err.into_service_error();
            let meta = err.meta();
            let message = meta.message().unwrap_or("unknown");
            let code = meta.code().unwrap_or("unknown");
            eprintln!("Error listing EC2 Security Groups: ({})) {}", code, message);
        }
    }
}
```

- API-Einheiten finden Sie [DescribeSecurityGroups](#) in der AWS SDK API Rust-Referenz.

Codebeispiele

- [Grundlegende Beispiele für die EC2 Verwendung von Amazon AWS SDKs](#)
 - [Hallo Amazon EC2](#)
 - [Lernen Sie die Grundlagen von Amazon EC2 mit einem AWS SDK](#)
 - [Aktionen für Amazon bei EC2 der Verwendung AWS SDKs](#)
 - [Verwenden Sie AcceptVpcPeeringConnection mit einem CLI](#)
 - [Verwenden Sie es AllocateAddress mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie AllocateHosts mit einem CLI](#)
 - [Verwenden Sie AssignPrivateIpAddresses mit einem CLI](#)
 - [Verwenden Sie es AssociateAddress mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie AssociateDhcpOptions mit einem CLI](#)
 - [Verwenden Sie AssociateRouteTable mit einem CLI](#)
 - [Verwenden Sie AttachInternetGateway mit einem CLI](#)
 - [Verwenden Sie AttachNetworkInterface mit einem CLI](#)
 - [Verwenden Sie AttachVolume mit einem CLI](#)
 - [Verwenden Sie AttachVpnGateway mit einem CLI](#)
 - [Verwenden Sie AuthorizeSecurityGroupEgress mit einem CLI](#)
 - [Verwenden Sie es AuthorizeSecurityGroupIngress mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie CancelCapacityReservation mit einem CLI](#)
 - [Verwenden Sie CancellImportTask mit einem CLI](#)
 - [Verwenden Sie CancelSpotFleetRequests mit einem CLI](#)
 - [Verwenden Sie CancelSpotInstanceRequests mit einem CLI](#)
 - [Verwenden Sie ConfirmProductInstance mit einem CLI](#)
 - [Verwenden Sie CopyImage mit einem CLI](#)
 - [Verwenden Sie CopySnapshot mit einem CLI](#)
 - [Verwenden Sie CreateCapacityReservation mit einem CLI](#)
 - [Verwenden Sie CreateCustomerGateway mit einem CLI](#)
 - [Verwenden Sie CreateDhcpOptions mit einem CLI](#)
 - [Verwenden Sie CreateFlowLogs mit einem CLI](#)
 - [Verwenden Sie CreateImage mit einem CLI](#)

- [Verwenden Sie CreateInstanceExportTask mit einem CLI](#)
- [Verwenden Sie CreateInternetGateway mit einem CLI](#)
- [Verwenden Sie es CreateKeyPair mit einem AWS SDK oder CLI](#)
- [Verwenden Sie es CreateLaunchTemplate mit einem AWS SDK oder CLI](#)
- [Verwenden Sie CreateNetworkAcl mit einem CLI](#)
- [Verwenden Sie CreateNetworkAclEntry mit einem CLI](#)
- [Verwenden Sie CreateNetworkInterface mit einem CLI](#)
- [Verwenden Sie CreatePlacementGroup mit einem CLI](#)
- [Verwenden Sie CreateRoute mit einem CLI](#)
- [Verwenden Sie es CreateRouteTable mit einem AWS SDK oder CLI](#)
- [Verwenden Sie es CreateSecurityGroup mit einem AWS SDK oder CLI](#)
- [Verwenden Sie CreateSnapshot mit einem CLI](#)
- [Verwenden Sie CreateSpotDatafeedSubscription mit einem CLI](#)
- [Verwenden Sie es CreateSubnet mit einem AWS SDK oder CLI](#)
- [Verwenden Sie es CreateTags mit einem AWS SDK oder CLI](#)
- [Verwenden Sie CreateVolume mit einem CLI](#)
- [Verwenden Sie es CreateVpc mit einem AWS SDK oder CLI](#)
- [Verwenden Sie es CreateVpcEndpoint mit einem AWS SDK oder CLI](#)
- [Verwenden Sie CreateVpnConnection mit einem CLI](#)
- [Verwenden Sie CreateVpnConnectionRoute mit einem CLI](#)
- [Verwenden Sie CreateVpnGateway mit einem CLI](#)
- [Verwenden Sie DeleteCustomerGateway mit einem CLI](#)
- [Verwenden Sie DeleteDhcpOptions mit einem CLI](#)
- [Verwenden Sie DeleteFlowLogs mit einem CLI](#)
- [Verwenden Sie DeleteInternetGateway mit einem CLI](#)
- [Verwenden Sie es DeleteKeyPair mit einem AWS SDK oder CLI](#)
- [Verwenden Sie es DeleteLaunchTemplate mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DeleteNetworkAcl mit einem CLI](#)
- [Verwenden Sie DeleteNetworkAclEntry mit einem CLI](#)
- [Verwenden Sie DeleteNetworkInterface mit einem CLI](#)

- [Verwenden Sie DeletePlacementGroup mit einem CLI](#)
- [Verwenden Sie DeleteRoute mit einem CLI](#)
- [Verwenden Sie DeleteRouteTable mit einem CLI](#)
- [Verwenden Sie es DeleteSecurityGroup mit einem AWS SDK oder CLI](#)
- [Verwenden Sie es DeleteSnapshot mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DeleteSpotDatafeedSubscription mit einem CLI](#)
- [Verwenden Sie DeleteSubnet mit einem CLI](#)
- [Verwenden Sie DeleteTags mit einem CLI](#)
- [Verwenden Sie DeleteVolume mit einem CLI](#)
- [Verwenden Sie es DeleteVpc mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DeleteVpcEndpoint mit einem AWS SDK](#)
- [Verwenden Sie DeleteVpnConnection mit einem CLI](#)
- [Verwenden Sie DeleteVpnConnectionRoute mit einem CLI](#)
- [Verwenden Sie DeleteVpnGateway mit einem CLI](#)
- [Verwenden Sie DeregisterImage mit einem CLI](#)
- [Verwenden Sie DescribeAccountAttributes mit einem CLI](#)
- [Verwenden Sie DescribeAddresses mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeAvailabilityZones mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeBundleTasks mit einem CLI](#)
- [Verwenden Sie DescribeCapacityReservations mit einem CLI](#)
- [Verwenden Sie DescribeCustomerGateways mit einem CLI](#)
- [Verwenden Sie DescribeDhcpOptions mit einem CLI](#)
- [Verwenden Sie DescribeFlowLogs mit einem CLI](#)
- [Verwenden Sie DescribeHostReservationOfferings mit einem CLI](#)
- [Verwenden Sie DescribeHosts mit einem CLI](#)
- [Verwenden Sie DescribeIamInstanceProfileAssociations mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeIdFormat mit einem CLI](#)
- [Verwenden Sie DescribeIdentityIdFormat mit einem CLI](#)
- [Verwenden Sie DescribeImageAttribute mit einem CLI](#)
- [Verwenden Sie DescribeImages mit einem AWS SDK oder CLI](#)

- [Verwenden Sie DescribeImportImageTasks mit einem CLI](#)
- [Verwenden Sie DescribeImportSnapshotTasks mit einem CLI](#)
- [Verwenden Sie DescribeInstanceAttribute mit einem CLI](#)
- [Verwenden Sie DescribeInstanceStatus mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeInstanceTypes mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeInstances mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeInternetGateways mit einem CLI](#)
- [Verwenden Sie DescribeKeyPairs mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeNetworkAcls mit einem CLI](#)
- [Verwenden Sie DescribeNetworkInterfaceAttribute mit einem CLI](#)
- [Verwenden Sie DescribeNetworkInterfaces mit einem CLI](#)
- [Verwenden Sie DescribePlacementGroups mit einem CLI](#)
- [Verwenden Sie DescribePrefixLists mit einem CLI](#)
- [Verwenden Sie DescribeRegions mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeRouteTables mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeScheduledInstanceAvailability mit einem CLI](#)
- [Verwenden Sie DescribeScheduledInstances mit einem CLI](#)
- [Verwenden Sie DescribeSecurityGroups mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeSnapshotAttribute mit einem CLI](#)
- [Verwenden Sie DescribeSnapshots mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeSpotDatafeedSubscription mit einem CLI](#)
- [Verwenden Sie DescribeSpotFleetInstances mit einem CLI](#)
- [Verwenden Sie DescribeSpotFleetRequestHistory mit einem CLI](#)
- [Verwenden Sie DescribeSpotFleetRequests mit einem CLI](#)
- [Verwenden Sie DescribeSpotInstanceRequests mit einem CLI](#)
- [Verwenden Sie DescribeSpotPriceHistory mit einem CLI](#)
- [Verwenden Sie DescribeSubnets mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeTags mit einem CLI](#)
- [Verwenden Sie DescribeVolumeAttribute mit einem CLI](#)
- [Verwenden Sie DescribeVolumeStatus mit einem CLI](#)

- [Verwenden Sie DescribeVolumes mit einem CLI](#)
- [Verwenden Sie DescribeVpcAttribute mit einem CLI](#)
- [Verwenden Sie DescribeVpcClassicLink mit einem CLI](#)
- [Verwenden Sie DescribeVpcClassicLinkDnsSupport mit einem CLI](#)
- [Verwenden Sie DescribeVpcEndpointServices mit einem CLI](#)
- [Verwenden Sie DescribeVpcEndpoints mit einem CLI](#)
- [Verwenden Sie DescribeVpcs mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeVpnConnections mit einem CLI](#)
- [Verwenden Sie DescribeVpnGateways mit einem CLI](#)
- [Verwenden Sie DetachInternetGateway mit einem CLI](#)
- [Verwenden Sie DetachNetworkInterface mit einem CLI](#)
- [Verwenden Sie DetachVolume mit einem CLI](#)
- [Verwenden Sie DetachVpnGateway mit einem CLI](#)
- [Verwenden Sie DisableVgwRoutePropagation mit einem CLI](#)
- [Verwenden Sie DisableVpcClassicLink mit einem CLI](#)
- [Verwenden Sie DisableVpcClassicLinkDnsSupport mit einem CLI](#)
- [Verwenden Sie DisassociateAddress mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DisassociateRouteTable mit einem CLI](#)
- [Verwenden Sie EnableVgwRoutePropagation mit einem CLI](#)
- [Verwenden Sie EnableVolumelo mit einem CLI](#)
- [Verwenden Sie EnableVpcClassicLink mit einem CLI](#)
- [Verwenden Sie EnableVpcClassicLinkDnsSupport mit einem CLI](#)
- [Verwenden Sie GetConsoleOutput mit einem CLI](#)
- [Verwenden Sie GetHostReservationPurchasePreview mit einem CLI](#)
- [Verwenden Sie GetPasswordData mit einem AWS SDK oder CLI](#)
- [Verwenden Sie ImportImage mit einem CLI](#)
- [Verwenden Sie ImportKeyPair mit einem CLI](#)
- [Verwenden Sie ImportSnapshot mit einem CLI](#)
- [Verwenden Sie ModifyCapacityReservation mit einem CLI](#)
- [Verwenden Sie ModifyHosts mit einem CLI](#)

- [Verwenden Sie ModifyIdFormat mit einem CLI](#)
- [Verwenden Sie ModifyImageAttribute mit einem CLI](#)
- [Verwenden Sie ModifyInstanceAttribute mit einem CLI](#)
- [Verwenden Sie ModifyInstanceCreditSpecification mit einem CLI](#)
- [Verwenden Sie ModifyNetworkInterfaceAttribute mit einem CLI](#)
- [Verwenden Sie ModifyReservedInstances mit einem CLI](#)
- [Verwenden Sie ModifySnapshotAttribute mit einem CLI](#)
- [Verwenden Sie ModifySpotFleetRequest mit einem CLI](#)
- [Verwenden Sie ModifySubnetAttribute mit einem CLI](#)
- [Verwenden Sie ModifyVolumeAttribute mit einem CLI](#)
- [Verwenden Sie ModifyVpcAttribute mit einem CLI](#)
- [Verwenden Sie MonitorInstances mit einem AWS SDK oder CLI](#)
- [Verwenden Sie MoveAddressToVpc mit einem CLI](#)
- [Verwenden Sie PurchaseHostReservation mit einem CLI](#)
- [Verwenden Sie PurchaseScheduledInstances mit einem CLI](#)
- [Verwenden Sie RebootInstances mit einem AWS SDK oder CLI](#)
- [Verwenden Sie RegisterImage mit einem CLI](#)
- [Verwenden Sie RejectVpcPeeringConnection mit einem CLI](#)
- [Verwenden Sie ReleaseAddress mit einem AWS SDK oder CLI](#)
- [Verwenden Sie ReleaseHosts mit einem CLI](#)
- [Verwenden Sie ReplacelamInstanceProfileAssociation mit einem AWS SDK oder CLI](#)
- [Verwenden Sie ReplaceNetworkAclAssociation mit einem CLI](#)
- [Verwenden Sie ReplaceNetworkAclEntry mit einem CLI](#)
- [Verwenden Sie ReplaceRoute mit einem CLI](#)
- [Verwenden Sie ReplaceRouteTableAssociation mit einem CLI](#)
- [Verwenden Sie ReportInstanceStatus mit einem CLI](#)
- [Verwenden Sie RequestSpotFleet mit einem CLI](#)
- [Verwenden Sie RequestSpotInstances mit einem CLI](#)
- [Verwenden Sie ResetImageAttribute mit einem CLI](#)
- [Verwenden Sie ResetInstanceAttribute mit einem CLI](#)

- [Verwenden Sie ResetNetworkInterfaceAttribute mit einem CLI](#)
 - [Verwenden Sie ResetSnapshotAttribute mit einem CLI](#)
 - [Verwenden Sie RevokeSecurityGroupEgress mit einem CLI](#)
 - [Verwenden Sie RevokeSecurityGroupIngress mit einem CLI](#)
 - [Verwenden Sie RunInstances mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie RunScheduledInstances mit einem CLI](#)
 - [Verwenden Sie StartInstances mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie StopInstances mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie TerminateInstances mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie UnassignPrivateIpAddresses mit einem CLI](#)
 - [Verwenden Sie UnmonitorInstances mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie UpdateSecurityGroupRuleDescriptionsIngress mit einem CLI](#)
- [Szenarien für die EC2 Verwendung von Amazon AWS SDKs](#)
 - [Erstellen und verwalten Sie einen ausfallsicheren Service mithilfe eines AWS SDK](#)

Grundlegende Beispiele für die EC2 Verwendung von Amazon AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie die Grundlagen von Amazon Elastic Compute Cloud mit verwenden können AWS SDKs.

Beispiele

- [Hallo Amazon EC2](#)
- [Lernen Sie die Grundlagen von Amazon EC2 mit einem AWS SDK](#)
- [Aktionen für Amazon bei EC2 der Verwendung AWS SDKs](#)
 - [Verwenden Sie AcceptVpcPeeringConnection mit einem CLI](#)
 - [Verwenden Sie es AllocateAddress mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie AllocateHosts mit einem CLI](#)
 - [Verwenden Sie AssignPrivateIpAddresses mit einem CLI](#)
 - [Verwenden Sie es AssociateAddress mit einem AWS SDK oder CLI](#)
- [Verwenden Sie AssociateDhcpOptions mit einem CLI](#)

- [Verwenden Sie AssociateRouteTable mit einem CLI](#)
- [Verwenden Sie AttachInternetGateway mit einem CLI](#)
- [Verwenden Sie AttachNetworkInterface mit einem CLI](#)
- [Verwenden Sie AttachVolume mit einem CLI](#)
- [Verwenden Sie AttachVpnGateway mit einem CLI](#)
- [Verwenden Sie AuthorizeSecurityGroupEgress mit einem CLI](#)
- [Verwenden Sie es AuthorizeSecurityGroupIngress mit einem AWS SDK oder CLI](#)
- [Verwenden Sie CancelCapacityReservation mit einem CLI](#)
- [Verwenden Sie CancellImportTask mit einem CLI](#)
- [Verwenden Sie CancelSpotFleetRequests mit einem CLI](#)
- [Verwenden Sie CancelSpotInstanceRequests mit einem CLI](#)
- [Verwenden Sie ConfirmProductInstance mit einem CLI](#)
- [Verwenden Sie CopyImage mit einem CLI](#)
- [Verwenden Sie CopySnapshot mit einem CLI](#)
- [Verwenden Sie CreateCapacityReservation mit einem CLI](#)
- [Verwenden Sie CreateCustomerGateway mit einem CLI](#)
- [Verwenden Sie CreateDhcpOptions mit einem CLI](#)
- [Verwenden Sie CreateFlowLogs mit einem CLI](#)
- [Verwenden Sie CreateImage mit einem CLI](#)
- [Verwenden Sie CreateInstanceExportTask mit einem CLI](#)
- [Verwenden Sie CreateInternetGateway mit einem CLI](#)
- [Verwenden Sie es CreateKeyPair mit einem AWS SDK oder CLI](#)
- [Verwenden Sie es CreateLaunchTemplate mit einem AWS SDK oder CLI](#)
- [Verwenden Sie CreateNetworkAcl mit einem CLI](#)
- [Verwenden Sie CreateNetworkAclEntry mit einem CLI](#)
- [Verwenden Sie CreateNetworkInterface mit einem CLI](#)
- [Verwenden Sie CreatePlacementGroup mit einem CLI](#)
- [Verwenden Sie CreateRoute mit einem CLI](#)
- [Verwenden Sie es CreateRouteTable mit einem AWS SDK oder CLI](#)
- [Verwenden Sie es CreateSecurityGroup mit einem AWS SDK oder CLI](#)

- [Verwenden Sie CreateSnapshot mit einem CLI](#)
 - [Verwenden Sie CreateSpotDatafeedSubscription mit einem CLI](#)
 - [Verwenden Sie es CreateSubnet mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie es CreateTags mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie CreateVolume mit einem CLI](#)
 - [Verwenden Sie es CreateVpc mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie es CreateVpcEndpoint mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie CreateVpnConnection mit einem CLI](#)
 - [Verwenden Sie CreateVpnConnectionRoute mit einem CLI](#)
 - [Verwenden Sie CreateVpnGateway mit einem CLI](#)
 - [Verwenden Sie DeleteCustomerGateway mit einem CLI](#)
 - [Verwenden Sie DeleteDhcpOptions mit einem CLI](#)
 - [Verwenden Sie DeleteFlowLogs mit einem CLI](#)
 - [Verwenden Sie DeleteInternetGateway mit einem CLI](#)
 - [Verwenden Sie es DeleteKeyPair mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie es DeleteLaunchTemplate mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie DeleteNetworkAcl mit einem CLI](#)
 - [Verwenden Sie DeleteNetworkAclEntry mit einem CLI](#)
 - [Verwenden Sie DeleteNetworkInterface mit einem CLI](#)
 - [Verwenden Sie DeletePlacementGroup mit einem CLI](#)
 - [Verwenden Sie DeleteRoute mit einem CLI](#)
 - [Verwenden Sie DeleteRouteTable mit einem CLI](#)
 - [Verwenden Sie es DeleteSecurityGroup mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie es DeleteSnapshot mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie DeleteSpotDatafeedSubscription mit einem CLI](#)
 - [Verwenden Sie DeleteSubnet mit einem CLI](#)
 - [Verwenden Sie DeleteTags mit einem CLI](#)
 - [Verwenden Sie DeleteVolume mit einem CLI](#)
 - [Verwenden Sie es DeleteVpc mit einem AWS SDK oder CLI](#)
-
- [Verwenden Sie DeleteVpcEndpoint mit einem AWS SDK](#)

- [Verwenden Sie DeleteVpnConnection mit einem CLI](#)
- [Verwenden Sie DeleteVpnConnectionRoute mit einem CLI](#)
- [Verwenden Sie DeleteVpnGateway mit einem CLI](#)
- [Verwenden Sie DeregisterImage mit einem CLI](#)
- [Verwenden Sie DescribeAccountAttributes mit einem CLI](#)
- [Verwenden Sie DescribeAddresses mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeAvailabilityZones mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeBundleTasks mit einem CLI](#)
- [Verwenden Sie DescribeCapacityReservations mit einem CLI](#)
- [Verwenden Sie DescribeCustomerGateways mit einem CLI](#)
- [Verwenden Sie DescribeDhcpOptions mit einem CLI](#)
- [Verwenden Sie DescribeFlowLogs mit einem CLI](#)
- [Verwenden Sie DescribeHostReservationOfferings mit einem CLI](#)
- [Verwenden Sie DescribeHosts mit einem CLI](#)
- [Verwenden Sie DescribeIamInstanceProfileAssociations mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeIdFormat mit einem CLI](#)
- [Verwenden Sie DescribeIdentityIdFormat mit einem CLI](#)
- [Verwenden Sie DescribeImageAttribute mit einem CLI](#)
- [Verwenden Sie DescribeImages mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeImportImageTasks mit einem CLI](#)
- [Verwenden Sie DescribeImportSnapshotTasks mit einem CLI](#)
- [Verwenden Sie DescribeInstanceAttribute mit einem CLI](#)
- [Verwenden Sie DescribeInstanceStatus mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeInstanceTypes mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeInstances mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeInternetGateways mit einem CLI](#)
- [Verwenden Sie DescribeKeyPairs mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeNetworkAcls mit einem CLI](#)
- [Verwenden Sie DescribeNetworkInterfaceAttribute mit einem CLI](#)
- [Verwenden Sie DescribeNetworkInterfaces mit einem CLI](#)

- [Verwenden Sie DescribePlacementGroups mit einem CLI](#)
- [Verwenden Sie DescribePrefixLists mit einem CLI](#)
- [Verwenden Sie DescribeRegions mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeRouteTables mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeScheduledInstanceAvailability mit einem CLI](#)
- [Verwenden Sie DescribeScheduledInstances mit einem CLI](#)
- [Verwenden Sie DescribeSecurityGroups mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeSnapshotAttribute mit einem CLI](#)
- [Verwenden Sie DescribeSnapshots mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeSpotDatafeedSubscription mit einem CLI](#)
- [Verwenden Sie DescribeSpotFleetInstances mit einem CLI](#)
- [Verwenden Sie DescribeSpotFleetRequestHistory mit einem CLI](#)
- [Verwenden Sie DescribeSpotFleetRequests mit einem CLI](#)
- [Verwenden Sie DescribeSpotInstanceRequests mit einem CLI](#)
- [Verwenden Sie DescribeSpotPriceHistory mit einem CLI](#)
- [Verwenden Sie DescribeSubnets mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeTags mit einem CLI](#)
- [Verwenden Sie DescribeVolumeAttribute mit einem CLI](#)
- [Verwenden Sie DescribeVolumeStatus mit einem CLI](#)
- [Verwenden Sie DescribeVolumes mit einem CLI](#)
- [Verwenden Sie DescribeVpcAttribute mit einem CLI](#)
- [Verwenden Sie DescribeVpcClassicLink mit einem CLI](#)
- [Verwenden Sie DescribeVpcClassicLinkDnsSupport mit einem CLI](#)
- [Verwenden Sie DescribeVpcEndpointServices mit einem CLI](#)
- [Verwenden Sie DescribeVpcEndpoints mit einem CLI](#)
- [Verwenden Sie DescribeVpcs mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeVpnConnections mit einem CLI](#)
- [Verwenden Sie DescribeVpnGateways mit einem CLI](#)
- [Verwenden Sie DetachInternetGateway mit einem CLI](#)
- [Verwenden Sie DetachNetworkInterface mit einem CLI](#)

- [Verwenden Sie DetachVolume mit einem CLI](#)
- [Verwenden Sie DetachVpnGateway mit einem CLI](#)
- [Verwenden Sie DisableVgwRoutePropagation mit einem CLI](#)
- [Verwenden Sie DisableVpcClassicLink mit einem CLI](#)
- [Verwenden Sie DisableVpcClassicLinkDnsSupport mit einem CLI](#)
- [Verwenden Sie DisassociateAddress mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DisassociateRouteTable mit einem CLI](#)
- [Verwenden Sie EnableVgwRoutePropagation mit einem CLI](#)
- [Verwenden Sie EnableVolumelo mit einem CLI](#)
- [Verwenden Sie EnableVpcClassicLink mit einem CLI](#)
- [Verwenden Sie EnableVpcClassicLinkDnsSupport mit einem CLI](#)
- [Verwenden Sie GetConsoleOutput mit einem CLI](#)
- [Verwenden Sie GetHostReservationPurchasePreview mit einem CLI](#)
- [Verwenden Sie GetPasswordData mit einem AWS SDK oder CLI](#)
- [Verwenden Sie ImportImage mit einem CLI](#)
- [Verwenden Sie ImportKeyPair mit einem CLI](#)
- [Verwenden Sie ImportSnapshot mit einem CLI](#)
- [Verwenden Sie ModifyCapacityReservation mit einem CLI](#)
- [Verwenden Sie ModifyHosts mit einem CLI](#)
- [Verwenden Sie ModifyIdFormat mit einem CLI](#)
- [Verwenden Sie ModifyImageAttribute mit einem CLI](#)
- [Verwenden Sie ModifyInstanceAttribute mit einem CLI](#)
- [Verwenden Sie ModifyInstanceCreditSpecification mit einem CLI](#)
- [Verwenden Sie ModifyNetworkInterfaceAttribute mit einem CLI](#)
- [Verwenden Sie ModifyReservedInstances mit einem CLI](#)
- [Verwenden Sie ModifySnapshotAttribute mit einem CLI](#)
- [Verwenden Sie ModifySpotFleetRequest mit einem CLI](#)
- [Verwenden Sie ModifySubnetAttribute mit einem CLI](#)
- [Verwenden Sie ModifyVolumeAttribute mit einem CLI](#)
- [Verwenden Sie ModifyVpcAttribute mit einem CLI](#)

- [Verwenden Sie MonitorInstances mit einem AWS SDK oder CLI](#)
- [Verwenden Sie MoveAddressToVpc mit einem CLI](#)
- [Verwenden Sie PurchaseHostReservation mit einem CLI](#)
- [Verwenden Sie PurchaseScheduledInstances mit einem CLI](#)
- [Verwenden Sie RebootInstances mit einem AWS SDK oder CLI](#)
- [Verwenden Sie RegisterImage mit einem CLI](#)
- [Verwenden Sie RejectVpcPeeringConnection mit einem CLI](#)
- [Verwenden Sie ReleaseAddress mit einem AWS SDK oder CLI](#)
- [Verwenden Sie ReleaseHosts mit einem CLI](#)
- [Verwenden Sie ReplacelamInstanceProfileAssociation mit einem AWS SDK oder CLI](#)
- [Verwenden Sie ReplaceNetworkAclAssociation mit einem CLI](#)
- [Verwenden Sie ReplaceNetworkAclEntry mit einem CLI](#)
- [Verwenden Sie ReplaceRoute mit einem CLI](#)
- [Verwenden Sie ReplaceRouteTableAssociation mit einem CLI](#)
- [Verwenden Sie ReportInstanceStatus mit einem CLI](#)
- [Verwenden Sie RequestSpotFleet mit einem CLI](#)
- [Verwenden Sie RequestSpotInstances mit einem CLI](#)
- [Verwenden Sie ResetImageAttribute mit einem CLI](#)
- [Verwenden Sie ResetInstanceAttribute mit einem CLI](#)
- [Verwenden Sie ResetNetworkInterfaceAttribute mit einem CLI](#)
- [Verwenden Sie ResetSnapshotAttribute mit einem CLI](#)
- [Verwenden Sie RevokeSecurityGroupEgress mit einem CLI](#)
- [Verwenden Sie RevokeSecurityGroupIngress mit einem CLI](#)
- [Verwenden Sie RunInstances mit einem AWS SDK oder CLI](#)
- [Verwenden Sie RunScheduledInstances mit einem CLI](#)
- [Verwenden Sie StartInstances mit einem AWS SDK oder CLI](#)
- [Verwenden Sie StopInstances mit einem AWS SDK oder CLI](#)
- [Verwenden Sie TerminateInstances mit einem AWS SDK oder CLI](#)
- [Verwenden Sie UnassignPrivateIpAddresses mit einem CLI](#)
- [Verwenden Sie UnmonitorInstances mit einem AWS SDK oder CLI](#)

- [Verwenden Sie UpdateSecurityGroupRuleDescriptionsIngress mit einem CLI](#)

Hallo Amazon EC2

Die folgenden Codebeispiele zeigen, wie Sie mit Amazon beginnen können EC2.

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
namespace EC2Actions;

public class HelloEc2
{
    /// <summary>
    /// HelloEc2 lists the existing security groups for the default users.
    /// </summary>
    /// <param name="args">Command line arguments</param>
    /// <returns>Async task.</returns>
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Elastic Compute Cloud (Amazon
        EC2).
        using var host =
            Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
                .ConfigureServices((_, services) =>
                    services.AddAWSService<IAmazonEC2>()
                        .AddTransient<EC2Wrapper>()
                )
                .Build();

        // Now the client is available for injection.
        var ec2Client = host.Services.GetRequiredService<IAmazonEC2>();
    }
}
```

```
try
{
    // Retrieve information for up to 10 Amazon EC2 security groups.
    var request = new DescribeSecurityGroupsRequest { MaxResults = 10, };
    var securityGroups = new List<SecurityGroup>();

    var paginatorForSecurityGroups =
        ec2Client.Paginators.DescribeSecurityGroups(request);

    await foreach (var securityGroup in
paginatorForSecurityGroups.SecurityGroups)
    {
        securityGroups.Add(securityGroup);
    }

    // Now print the security groups returned by the call to
    // DescribeSecurityGroupsAsync.
    Console.WriteLine("Welcome to the EC2 Hello Service example. " +
        "\nLet's list your Security Groups:");
    securityGroups.ForEach(group =>
    {
        Console.WriteLine(
            $"Security group: {group.GroupName} ID: {group.GroupId}");
    });
}
catch (AmazonEC2Exception ex)
{
    Console.WriteLine($"An Amazon EC2 service error occurred while
listing security groups. {ex.Message}");
}
catch (Exception ex)
{
    Console.WriteLine($"An error occurred while listing security groups.
{ex.Message}");
}
}
```

- API-Einheiten finden Sie [DescribeSecurityGroups](#) unter AWS SDK for .NET API-Referenz.

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Code für die CMakeLists CMake .txt-Datei.

```
# 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 ec2)

# Set this project's name.
project("hello_ec2")

# 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_ec2.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})
```

Code für die Quelldatei hello_ec2.cpp.

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeInstancesRequest.h>
#include <iomanip>
#include <iostream>

/*
 * A "Hello EC2" starter application which initializes an Amazon Elastic Compute
 * Cloud (Amazon EC2) client and describes
 * the Amazon EC2 instances.
 *
 * main function
 *
 * Usage: 'hello_ec2'
 *
 */

int main(int argc, char **argv) {
    (void)argc;
    (void)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";

    Aws::EC2::EC2Client ec2Client(clientConfig);
    Aws::EC2::Model::DescribeInstancesRequest request;
    bool header = false;
    bool done = false;
    while (!done) {
        Aws::EC2::Model::DescribeInstancesOutcome outcome =
ec2Client.DescribeInstances(request);
        if (outcome.IsSuccess()) {
            if (!header) {
                std::cout << std::left <<
                    std::setw(48) << "Name" <<
                    std::setw(20) << "ID" <<
                    std::setw(25) << "Ami" <<
                    std::setw(15) << "Type" <<
                    std::setw(15) << "State" <<
                    std::setw(15) << "Monitoring" << std::endl;
                header = true;
            }

            const std::vector<Aws::EC2::Model::Reservation> &reservations =
                outcome.GetResult().GetReservations();

            for (const auto &reservation: reservations) {
                const std::vector<Aws::EC2::Model::Instance> &instances =
                    reservation.GetInstances();
                for (const auto &instance: instances) {
                    Aws::String instanceStateString =

Aws::EC2::Model::InstanceStateNameMapper::GetNameForInstanceStateName(
                        instance.GetState().GetName());

                    Aws::String typeString =

Aws::EC2::Model::InstanceTypeMapper::GetNameForInstanceType(
                        instance.GetInstanceType());

                    Aws::String monitorString =
```

```
Aws::EC2::Model::MonitoringStateMapper::GetNameForMonitoringState(
    instance.GetMonitoring().GetState());
    Aws::String name = "Unknown";

    const std::vector<Aws::EC2::Model::Tag> &tags =
instance.GetTags();
    auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
        [](const
Aws::EC2::Model::Tag &tag) {
    return tag.GetKey() ==
    "Name";
    });
    if (nameIter != tags.cend()) {
        name = nameIter->GetValue();
    }
    std::cout <<
        std::setw(48) << name <<
        std::setw(20) << instance.GetInstanceId() <<
        std::setw(25) << instance.GetImageId() <<
        std::setw(15) << typeString <<
        std::setw(15) << instanceStateString <<
        std::setw(15) << monitorString << std::endl;
    }
}

if (!outcome.GetResult().GetNextToken().empty()) {
    request.SetNextToken(outcome.GetResult().GetNextToken());
} else {
    done = true;
}
} else {
    std::cerr << "Failed to describe EC2 instances:" <<
        outcome.GetError().GetMessage() << std::endl;
    result = 1;
    break;
}
}
}

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

- API-Einheiten finden Sie unter [DescribeSecurityGroups AWS SDK for C++ API-Referenz](#).

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu. [Auf GitHub](#) sehen Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Asynchronously describes the security groups for the specified group ID.
 *
 * @param groupName the name of the security group to describe
 * @return a {@link CompletableFuture} that represents the asynchronous
operation
 *         of describing the security groups. The future will complete with a
 *         {@link DescribeSecurityGroupsResponse} object that contains the
 *         security group information.
 */
public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
    DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
        .groupNames(groupName)
        .build();

    DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
    AtomicReference<String> groupIdRef = new AtomicReference<>();
    return paginator.subscribe(response -> {
        response.securityGroups().stream()
            .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
            .findFirst()
            .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
    });
}
```

```
    }).thenApply(v -> {
        String groupId = groupIdRef.get();
        if (groupId == null) {
            throw new RuntimeException("No security group found with the
name: " + groupName);
        }
        return groupId;
    }).exceptionally(ex -> {
        logger.info("Failed to describe security group: " + ex.getMessage());
        throw new RuntimeException("Failed to describe security group", ex);
    });
}
```

- API Einzelheiten finden Sie [DescribeSecurityGroups](#) unter AWS SDK for Java 2.x API Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import { DescribeSecurityGroupsCommand, EC2Client } from "@aws-sdk/client-ec2";

// Call DescribeSecurityGroups and display the result.
export const main = async () => {
    const client = new EC2Client();
    try {
        const { SecurityGroups } = await client.send(
            new DescribeSecurityGroupsCommand({}),
        );

        const securityGroupList = SecurityGroups.slice(0, 9)
            .map((sg) => ` • ${sg.GroupId}: ${sg.GroupName}`)
            .join("\n");
    }
}
```

```
    console.log(
      "Hello, Amazon EC2! Let's list up to 10 of your security groups:",
    );
    console.log(securityGroupList);
  } catch (err) {
    console.error(err);
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- API-Einheiten finden Sie [DescribeSecurityGroups](#) unter AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
suspend fun describeEC2SecurityGroups(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->

        val response = ec2.describeSecurityGroups(request)
        response.securityGroups?.forEach { group ->
```

```

        println("Found Security Group with id ${group.groupId}, vpc id
        ${group.vpcId} and description ${group.description}")
    }
}
}

```

- API-Einheiten finden Sie [DescribeSecurityGroups](#) in der AWS SDK-API-Kotlin-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

def hello_ec2(ec2_client):
    """
    Use the AWS SDK for Python (Boto3) to list the security groups in your
    account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ec2_client: A Boto3 EC2 client. This client provides low-level
        access to AWS EC2 services.
    """
    print("Hello, Amazon EC2! Let's list up to 10 of your security groups:")
    try:
        paginator = ec2_client.get_paginator("describe_security_groups")
        response_iterator = paginator.paginate(MaxResults=10)
        for page in response_iterator:
            for sg in page["SecurityGroups"]:
                logger.info(f"\t{sg['GroupId']}: {sg['GroupName']}")
    except ClientError as err:
        logger.error("Failed to list security groups.")
        if err.response["Error"]["Code"] == "AccessDeniedException":
            logger.error("You do not have permission to list security groups.")
            raise

```

```
if __name__ == "__main__":
    hello_ec2(boto3.client("ec2"))
```

- API-Einheiten finden Sie unter [DescribeSecurityGroups](#) Python (Boto3) API - Referenz.AWS SDK

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
require 'aws-sdk-ec2'
require 'logger'

# EC2Manager is a class responsible for managing EC2 operations
# such as listing all EC2 instances in the current AWS account.
class EC2Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all EC2 instances in the current AWS account.
  def list_instances
    @logger.info('Listing instances')

    instances = fetch_instances

    if instances.empty?
      @logger.info('You have no instances')
    else
      print_instances(instances)
    end
  end
end
```



```
end

private

# Fetches all EC2 instances using pagination.
#
# @return [Array<Aws::EC2::Types::Instance>] List of EC2 instances.
def fetch_instances
  paginator = @client.describe_instances
  instances = []

  paginator.each_page do |page|
    page.reservations.each do |reservation|
      reservation.instances.each do |instance|
        instances << instance
      end
    end
  end

  instances
end

# Prints details of the given EC2 instances.
#
# @param instances [Array<Aws::EC2::Types::Instance>] List of EC2 instances to
print.
def print_instances(instances)
  instances.each do |instance|
    @logger.info("Instance ID: #{instance.instance_id}")
    @logger.info("Instance Type: #{instance.instance_type}")
    @logger.info("Public IP: #{instance.public_ip_address}")
    @logger.info("Public DNS Name: #{instance.public_dns_name}")
    @logger.info("\n")
  end
end

end

if $PROGRAM_NAME == __FILE__
  ec2_client = Aws::EC2::Client.new(region: 'us-west-2')
  manager = EC2Manager.new(ec2_client)
  manager.list_instances
end
```

- API-Einheiten finden Sie [DescribeSecurityGroups](#) unter AWS SDK for Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
async fn show_security_groups(client: &aws_sdk_ec2::Client, group_ids:
Vec<String>) {
    let response = client
        .describe_security_groups()
        .set_group_ids(Some(group_ids))
        .send()
        .await;

    match response {
        Ok(output) => {
            for group in output.security_groups() {
                println!(
                    "Found Security Group {} ({}), vpc id {} and description {}",
                    group.group_name().unwrap_or("unknown"),
                    group.group_id().unwrap_or("id-unknown"),
                    group.vpc_id().unwrap_or("vpcid-unknown"),
                    group.description().unwrap_or("(none)")
                );
            }
        }
        Err(err) => {
            let err = err.into_service_error();
            let meta = err.meta();
            let message = meta.message().unwrap_or("unknown");
            let code = meta.code().unwrap_or("unknown");
            eprintln!("Error listing EC2 Security Groups: ({})) {}", code, message);
        }
    }
}
```

```
}
```

- API-Einheiten finden Sie [DescribeSecurityGroups](#) in der AWS SDK-API-Rust-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Lernen Sie die Grundlagen von Amazon EC2 mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie ein Schlüsselpaar und eine Sicherheitsgruppe.
- Wählen Sie ein Amazon Machine Image (AMI) und einen kompatiblen Instance-Typ aus und erstellen Sie dann eine Instance.
- Halten Sie die Instance an und starten Sie sie neu.
- Verknüpfen einer Elastic-IP-Adresse mit der Instance.
- Connect zu Ihrer Instance mit her SSH und bereinigen Sie anschließend die Ressourcen.

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Führen Sie ein Szenario an einer Eingabeaufforderung aus.

```
/// <summary>
/// Show Amazon Elastic Compute Cloud (Amazon EC2) Basics actions.
/// </summary>
public class EC2Basics
{
    public static ILogger<EC2Basics> _logger = null!;
    public static EC2Wrapper _ec2Wrapper = null!;
```

```
public static SsmWrapper _ssmWrapper = null!;  
public static UiMethods _uiMethods = null!;  
  
public static string associationId = null!;  
public static string allocationId = null!;  
public static string instanceId = null!;  
public static string keyPairName = null!;  
public static string groupName = null!;  
public static string tempFileName = null!;  
public static string secGroupId = null!;  
public static bool isInteractive = true;  
  
/// <summary>  
/// Perform the actions defined for the Amazon EC2 Basics scenario.  
/// </summary>  
/// <param name="args">Command line arguments.</param>  
/// <returns>A Task object.</returns>  
public static async Task Main(string[] args)  
{  
    // Set up dependency injection for Amazon EC2 and Amazon Simple Systems  
    // Management (Amazon SSM) Service.  
    using var host =  
Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)  
        .ConfigureServices((_, services) =>  
            services.AddAWSService<IAmazonEC2>()  
                .AddAWSService<IAmazonSimpleSystemsManagement>()  
                .AddTransient<EC2Wrapper>()  
                .AddTransient<SsmWrapper>()  
        )  
        .Build();  
  
    SetUpServices(host);  
  
    var uniqueName = Guid.NewGuid().ToString();  
    keyPairName = "mvp-example-key-pair" + uniqueName;  
    groupName = "ec2-scenario-group" + uniqueName;  
    var groupDescription = "A security group created for the EC2 Basics  
scenario.";  
  
    try  
    {  
        // Start the scenario.  
        _uiMethods.DisplayOverview();  
        _uiMethods.PressEnter(isInteractive);  
    }  
}
```

```
// Create the key pair.
_uiMethods.DisplayTitle("Create RSA key pair");
Console.WriteLine("Let's create an RSA key pair that you can be use to
");

Console.WriteLine("securely connect to your EC2 instance.");
var keyPair = await _ec2Wrapper.CreateKeyPair(keyPairName);

// Save key pair information to a temporary file.
tempFileName = _ec2Wrapper.SaveKeyPair(keyPair);

Console.WriteLine(
    $"Created the key pair: {keyPair.KeyName} and saved it to:
{tempFileName}");
string? answer = "";
if (isInteractive)
{
    do
    {
        Console.WriteLine("Would you like to list your existing key
pairs? ");

        answer = Console.ReadLine();
    } while (answer!.ToLower() != "y" && answer.ToLower() != "n");
}

if (!isInteractive || answer == "y")
{
    // List existing key pairs.
    _uiMethods.DisplayTitle("Existing key pairs");

    // Passing an empty string to the DescribeKeyPairs method will
return
    // a list of all existing key pairs.
    var keyPairs = await _ec2Wrapper.DescribeKeyPairs("");
    keyPairs.ForEach(kp =>
    {
        Console.WriteLine(
            $"{kp.KeyName} created at: {kp.CreateTime} Fingerprint:
{kp.KeyFingerprint}");
    });
}

_uiMethods.PressEnter(isInteractive);
```

```
// Create the security group.
Console.WriteLine(
    "Let's create a security group to manage access to your
instance.");
    secGroupId = await _ec2Wrapper.CreateSecurityGroup(groupName,
groupDescription);
    Console.WriteLine(
        "Let's add rules to allow all HTTP and HTTPS inbound traffic and
to allow SSH only from your current IP address.");

    _uiMethods.DisplayTitle("Security group information");
    var secGroups = await _ec2Wrapper.DescribeSecurityGroups(secGroupId);

    Console.WriteLine($"Created security group {groupName} in your
default VPC.");
    secGroups.ForEach(group =>
    {
        _ec2Wrapper.DisplaySecurityGroupInfoAsync(group);
    });
    _uiMethods.PressEnter(isInteractive);

    Console.WriteLine(
        "Now we'll authorize the security group we just created so that
it can");
    Console.WriteLine("access the EC2 instances you create.");
    await _ec2Wrapper.AuthorizeSecurityGroupIngress(groupName);

    secGroups = await _ec2Wrapper.DescribeSecurityGroups(secGroupId);
    Console.WriteLine($"Now let's look at the permissions again.");
    secGroups.ForEach(group =>
    {
        _ec2Wrapper.DisplaySecurityGroupInfoAsync(group);
    });
    _uiMethods.PressEnter(isInteractive);

    // Get list of available Amazon Linux 2 Amazon Machine Images (AMIs).
    var parameters =
        await _ssmWrapper.GetParametersByPath(
            "/aws/service/ami-amazon-linux-latest");

    List<string> imageIds = parameters.Select(param =>
param.Value).ToList();

    var images = await _ec2Wrapper.DescribeImages(imageIds);
```

```
var i = 1;
images.ForEach(image =>
{
    Console.WriteLine($"{i++}\t{image.Description}");
});

int choice = 1;
bool validNumber = false;
if (isInteractive)
{
    do
    {
        Console.Write("Please select an image: ");
        var selImage = Console.ReadLine();
        validNumber = int.TryParse(selImage, out choice);
    } while (!validNumber);
}

var selectedImage = images[choice - 1];

// Display available instance types.
_uiMethods.DisplayTitle("Instance Types");
var instanceTypes =
    await
_ec2Wrapper.DescribeInstanceTypes(selectedImage.Architecture);

i = 1;
instanceTypes.ForEach(instanceType =>
{
    Console.WriteLine($"{i++}\t{instanceType.InstanceType}");
});
if (isInteractive)
{
    do
    {
        Console.Write("Please select an instance type: ");
        var selImage = Console.ReadLine();
        validNumber = int.TryParse(selImage, out choice);
    } while (!validNumber);
}

var selectedInstanceType = instanceTypes[choice - 1].InstanceType;
```

```
// Create an EC2 instance.
_uiMethods.DisplayTitle("Creating an EC2 Instance");
instanceId = await _ec2Wrapper.RunInstances(selectedImage.ImageId,
    selectedInstanceType, keyPairName, secGroupId);

_uiMethods.PressEnter(isInteractive);

var instance = await _ec2Wrapper.DescribeInstance(instanceId);
_uiMethods.DisplayTitle("New Instance Information");
_ec2Wrapper.DisplayInstanceInformation(instance);

Console.WriteLine(
    "\nYou can use SSH to connect to your instance. For example:");
Console.WriteLine(
    $"{"\tssh -i {tempFileName} ec2-user@{instance.PublicIpAddress}");

_uiMethods.PressEnter(isInteractive);

Console.WriteLine(
    "Now we'll stop the instance and then start it again to see
what's changed.");

await _ec2Wrapper.StopInstances(instanceId);

Console.WriteLine("Now let's start it up again.");
await _ec2Wrapper.StartInstances(instanceId);

Console.WriteLine("\nLet's see what changed.");

instance = await _ec2Wrapper.DescribeInstance(instanceId);
_uiMethods.DisplayTitle("New Instance Information");
_ec2Wrapper.DisplayInstanceInformation(instance);

Console.WriteLine("\nNotice the change in the SSH information:");
Console.WriteLine(
    $"{"\tssh -i {tempFileName} ec2-user@{instance.PublicIpAddress}");

_uiMethods.PressEnter(isInteractive);

Console.WriteLine(
    "Now we will stop the instance again. Then we will create and
associate an");
Console.WriteLine("Elastic IP address to use with our instance.");
```



```
        await _ec2Wrapper.StopInstances(instanceId);
        _uiMethods.PressEnter(isInteractive);

        _uiMethods.DisplayTitle("Allocate Elastic IP address");
        Console.WriteLine(
            "You can allocate an Elastic IP address and associate it
with your instance\nto keep a consistent IP address even when your instance
restarts.");
        var allocationResponse = await _ec2Wrapper.AllocateAddress();
        allocationId = allocationResponse.AllocationId;
        Console.WriteLine(
            "Now we will associate the Elastic IP address with our
instance.");
        associationId = await _ec2Wrapper.AssociateAddress(allocationId,
instanceId);

        // Start the instance again.
        Console.WriteLine("Now let's start the instance again.");
        await _ec2Wrapper.StartInstances(instanceId);

        Console.WriteLine("\nLet's see what changed.");

        instance = await _ec2Wrapper.DescribeInstance(instanceId);
        _uiMethods.DisplayTitle("Instance information");
        _ec2Wrapper.DisplayInstanceInformation(instance);

        Console.WriteLine("\nHere is the SSH information:");
        Console.WriteLine(
            $"{tssh -i {tempFileName} ec2-user@{instance.PublicIpAddress}");

        Console.WriteLine("Let's stop and start the instance again.");
        _uiMethods.PressEnter(isInteractive);

        await _ec2Wrapper.StopInstances(instanceId);

        Console.WriteLine("\nThe instance has stopped.");

        Console.WriteLine("Now let's start it up again.");
        await _ec2Wrapper.StartInstances(instanceId);

        instance = await _ec2Wrapper.DescribeInstance(instanceId);
        _uiMethods.DisplayTitle("New Instance Information");
        _ec2Wrapper.DisplayInstanceInformation(instance);
```

```
        Console.WriteLine("Note that the IP address did not change this
time.");
        _uiMethods.PressEnter(isInteractive);

        await Cleanup();
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "There was a problem with the scenario, starting
cleanup.");
        await Cleanup();
    }

    _uiMethods.DisplayTitle("EC2 Basics Scenario completed.");
    _uiMethods.PressEnter(isInteractive);
}

/// <summary>
/// Set up the services and logging.
/// </summary>
/// <param name="host"></param>
public static void SetupServices(IHost host)
{
    var loggerFactory = LoggerFactory.Create(builder =>
    {
        builder.AddConsole();
    });
    _logger = new Logger<EC2Basics>(loggerFactory);

    // Now the client is available for injection.
    _ec2Wrapper = host.Services.GetRequiredService<EC2Wrapper>();
    _ssmWrapper = host.Services.GetRequiredService<SsmWrapper>();
    _uiMethods = new UiMethods();
}

/// <summary>
/// Clean up any resources from the scenario.
/// </summary>
/// <returns></returns>
public static async Task Cleanup()
{
    _uiMethods.DisplayTitle("Clean up resources");
    Console.WriteLine("Now let's clean up the resources we created.");
}
```

```
Console.WriteLine("Disassociate the Elastic IP address and release it.");
// Disassociate the Elastic IP address.
await _ec2Wrapper.DisassociateIp(associationId);

// Delete the Elastic IP address.
await _ec2Wrapper.ReleaseAddress(allocationId);

// Terminate the instance.
Console.WriteLine("Terminating the instance we created.");
await _ec2Wrapper.TerminateInstances(instanceId);

// Delete the security group.
Console.WriteLine($"Deleting the Security Group: {groupName}.");
await _ec2Wrapper.DeleteSecurityGroup(secGroupId);

// Delete the RSA key pair.
Console.WriteLine($"Deleting the key pair: {keyPairName}");
await _ec2Wrapper.DeleteKeyPair(keyPairName);
Console.WriteLine("Deleting the temporary file with the key
information.");
    _ec2Wrapper.DeleteTempFile(tempFileName);
    _uiMethods.PressEnter(isInteractive);
}
}
```

Definiere eine Klasse, die EC2 Aktionen umschließt.

```
/// <summary>
/// Methods of this class perform Amazon Elastic Compute Cloud (Amazon EC2).
/// </summary>
public class EC2Wrapper
{
    private readonly IAmazonEC2 _amazonEC2;
    private readonly ILogger<EC2Wrapper> _logger;

    /// <summary>
    /// Constructor for the EC2Wrapper class.
    /// </summary>
    /// <param name="amazonScheduler">The injected EC2 client.</param>
    /// <param name="logger">The injected logger.</param>
    public EC2Wrapper(IAmazonEC2 amazonService, ILogger<EC2Wrapper> logger)
    {
```

```
        _amazonEC2 = amazonService;
        _logger = logger;
    }

    /// <summary>
    /// Allocates an Elastic IP address that can be associated with an Amazon EC2
    /// instance. By using an Elastic IP address, you can keep the public IP
address
    /// constant even when you restart the associated instance.
    /// </summary>
    /// <returns>The response object for the allocated address.</returns>
    public async Task<AllocateAddressResponse> AllocateAddress()
    {
        var request = new AllocateAddressRequest();

        try
        {
            var response = await _amazonEC2.AllocateAddressAsync(request);
            Console.WriteLine($"Allocated IP: {response.PublicIp} with allocation
ID {response.AllocationId}.");
            return response;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "AddressLimitExceeded")
            {
                // For more information on Elastic IP address quotas, see:
                // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-
ip-addresses-eip.html#using-instance-addressing-limit
                _logger.LogError($"Unable to allocate Elastic IP, address limit
exceeded. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while allocating Elastic IP.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
```

```
    /// Associates an Elastic IP address with an instance. When this association
    is
    /// created, the Elastic IP's public IP address is immediately used as the
    public
    /// IP address of the associated instance.
    /// </summary>
    /// <param name="allocationId">The allocation Id of an Elastic IP address.</
param>
    /// <param name="instanceId">The instance Id of the EC2 instance to
    /// associate the address with.</param>
    /// <returns>The association Id that represents
    /// the association of the Elastic IP address with an instance.</returns>
    public async Task<string> AssociateAddress(string allocationId, string
instanceId)
    {
        try
        {
            var request = new AssociateAddressRequest
            {
                AllocationId = allocationId,
                InstanceId = instanceId
            };

            var response = await _amazonEC2.AssociateAddressAsync(request);
            return response.AssociationId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceId")
            {
                _logger.LogError(
                    $"InstanceId is invalid, unable to associate address.
{ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while associating the Elastic IP.:
{ex.Message}");
            throw;
        }
    }
}
```

```
}

/// <summary>
/// Authorize the local computer ingress to EC2 instances associated
/// with the virtual private cloud (VPC) security group.
/// </summary>
/// <param name="groupName">The name of the security group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> AuthorizeSecurityGroupIngress(string groupName)
{
    try
    {
        // Get the IP address for the local computer.
        var ipAddress = await GetIpAddress();
        Console.WriteLine($"Your IP address is: {ipAddress}");
        var ipRanges =
            new List<IpRange> { new IpRange { CidrIp = $"{ipAddress}/32" } };
        var permission = new IpPermission
        {
            Ipv4Ranges = ipRanges,
            IpProtocol = "tcp",
            FromPort = 22,
            ToPort = 22
        };
        var permissions = new List<IpPermission> { permission };
        var response = await _amazonEC2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest(groupName,
permissions));
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidPermission.Duplicate")
        {
            _logger.LogError(
                $"The ingress rule already exists. {ec2Exception.Message}");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(
            $"An error occurred while authorizing ingress.: {ex.Message}");
    }
}
```

```
        throw;
    }
}

/// <summary>
/// Authorize the local computer for ingress to
/// the Amazon EC2 SecurityGroup.
/// </summary>
/// <returns>The IPv4 address of the computer running the scenario.</returns>
private static async Task<string> GetIpAddress()
{
    var httpClient = new HttpClient();
    var ipString = await httpClient.GetStringAsync("https://
checkip.amazonaws.com");

    // The IP address is returned with a new line
    // character on the end. Trim off the whitespace and
    // return the value to the caller.
    return ipString.Trim();
}

/// <summary>
/// Create an Amazon EC2 key pair with a specified name.
/// </summary>
/// <param name="keyPairName">The name for the new key pair.</param>
/// <returns>The Amazon EC2 key pair created.</returns>
public async Task<KeyPair?> CreateKeyPair(string keyPairName)
{
    try
    {
        var request = new CreateKeyPairRequest { KeyName = keyPairName, };

        var response = await _amazonEC2.CreateKeyPairAsync(request);

        var kp = response.KeyPair;
        // Return the key pair so it can be saved if needed.

        // Wait until the key pair exists.
        int retries = 5;
        while (retries-- > 0)
        {
            Console.WriteLine($"Checking for new KeyPair {keyPairName}...");
            var keyPairs = await DescribeKeyPairs(keyPairName);
            if (keyPairs.Any())
```

```
        {
            return kp;
        }

        Thread.Sleep(5000);
        retries--;
    }
    _logger.LogError($"Unable to find newly created KeyPair
{keyPairName}.");
    throw new DoesNotExistException("KeyPair not found");
}
catch (AmazonEC2Exception ec2Exception)
{
    if (ec2Exception.ErrorCode == "InvalidKeyPair.Duplicate")
    {
        _logger.LogError(
            $"A key pair called {keyPairName} already exists.");
    }

    throw;
}
catch (Exception ex)
{
    _logger.LogError(
        $"An error occurred while creating the key pair.: {ex.Message}");
    throw;
}
}

/// <summary>
/// Save KeyPair information to a temporary file.
/// </summary>
/// <param name="keyPair">The name of the key pair.</param>
/// <returns>The full path to the temporary file.</returns>
public string SaveKeyPair(KeyPair keyPair)
{
    var tempPath = Path.GetTempPath();
    var tempFileName = $"{tempPath}\\{Path.GetRandomFileName()}";
    var pemFileName = Path.ChangeExtension(tempFileName, "pem");

    // Save the key pair to a file in a temporary folder.
    using var stream = new FileStream(pemFileName, FileMode.Create);
    using var writer = new StreamWriter(stream);
    writer.WriteLine(keyPair.KeyMaterial);
}
```



```
        return pemFileName;
    }

    /// <summary>
    /// Create an Amazon EC2 security group with a specified name and
    description.
    /// </summary>
    /// <param name="groupName">The name for the new security group.</param>
    /// <param name="groupDescription">A description of the new security group.</
param>
    /// <returns>The group Id of the new security group.</returns>
    public async Task<string> CreateSecurityGroup(string groupName, string
groupDescription)
    {
        try
        {
            var response = await _amazonEC2.CreateSecurityGroupAsync(
                new CreateSecurityGroupRequest(groupName, groupDescription));

            // Wait until the security group exists.
            int retries = 5;
            while (retries-- > 0)
            {
                var groups = await DescribeSecurityGroups(response.GroupId);
                if (groups.Any())
                {
                    return response.GroupId;
                }

                Thread.Sleep(5000);
                retries--;
            }
            _logger.LogError($"Unable to find newly created group {groupName}.");
            throw new DoesNotExistException("security group not found");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "ResourceAlreadyExists")
            {
                _logger.LogError(
                    $"A security group with the name {groupName} already exists.
{ec2Exception.Message}");
            }
        }
    }
}
```

```
        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(
            $"An error occurred while creating the security group.:
{ex.Message}");
        throw;
    }
}

/// <summary>
/// Create a new Amazon EC2 VPC.
/// </summary>
/// <param name="cidrBlock">The CIDR block for the new security group.</
param>
/// <returns>The VPC Id of the new VPC.</returns>
public async Task<string?> CreateVPC(string cidrBlock)
{
    try
    {
        var response = await _amazonEC2.CreateVpcAsync(new CreateVpcRequest
        {
            CidrBlock = cidrBlock,
        });

        Vpc vpc = response.Vpc;
        Console.WriteLine($"Created VPC with ID: {vpc.VpcId}.");
        return vpc.VpcId;
    }
    catch (AmazonEC2Exception ex)
    {
        Console.WriteLine($"Couldn't create VPC because: {ex.Message}");
        return null;
    }
}

/// <summary>
/// Delete an Amazon EC2 key pair.
/// </summary>
/// <param name="keyPairName">The name of the key pair to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
```

```
public async Task<bool> DeleteKeyPair(string keyPairName)
{
    try
    {
        await _amazonEC2.DeleteKeyPairAsync(new
DeleteKeyPairRequest(keyPairName)).ConfigureAwait(false);
        return true;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
        {
            _logger.LogError($"KeyPair {keyPairName} does not exist and
cannot be deleted. Please verify the key pair name and try again.");
        }

        return false;
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Couldn't delete the key pair because:
{ex.Message}");
        return false;
    }
}

/// <summary>
/// Delete the temporary file where the key pair information was saved.
/// </summary>
/// <param name="tempFileName">The path to the temporary file.</param>
public void DeleteTempFile(string tempFileName)
{
    if (File.Exists(tempFileName))
    {
        File.Delete(tempFileName);
    }
}

/// <summary>
/// Delete an Amazon EC2 security group.
/// </summary>
/// <param name="groupName">The name of the group to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteSecurityGroup(string groupId)
```

```
{
    try
    {
        var response =
            await _amazonEC2.DeleteSecurityGroupAsync(
                new DeleteSecurityGroupRequest { GroupId = groupId });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
        {
            _logger.LogError(
                $"Security Group {groupId} does not exist and cannot be
deleted. Please verify the ID and try again.");
        }

        return false;
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
        return false;
    }
}

/// <summary>
/// Delete an Amazon EC2 VPC.
/// </summary>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteVpc(string vpcId)
{
    var request = new DeleteVpcRequest
    {
        VpcId = vpcId,
    };

    var response = await _amazonEC2.DeleteVpcAsync(request);

    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
```

```
/// Get information about existing Amazon EC2 images.
/// </summary>
/// <returns>A list of image information.</returns>
public async Task<List<Image>> DescribeImages(List<string>? imageIds)
{
    var request = new DescribeImagesRequest();
    if (imageIds is not null)
    {
        // If the imageIds list is not null, add the list
        // to the request object.
        request.ImageIds = imageIds;
    }

    var response = await _amazonEC2.DescribeImagesAsync(request);
    return response.Images;
}

/// <summary>
/// Display the information returned by DescribeImages.
/// </summary>
/// <param name="images">The list of image information to display.</param>
public void DisplayImageInfo(List<Image> images)
{
    images.ForEach(image =>
    {
        Console.WriteLine($"{image.Name} Created on: {image.CreationDate}");
    });
}

/// <summary>
/// Get information about an Amazon EC2 instance.
/// </summary>
/// <param name="instanceId">The instance Id of the EC2 instance.</param>
/// <returns>An EC2 instance.</returns>
public async Task<Instance> DescribeInstance(string instanceId)
{
    var response = await _amazonEC2.DescribeInstancesAsync(
        new DescribeInstancesRequest { InstanceIds = new List<string>
{ instanceId } });
    return response.Reservations[0].Instances[0];
}

/// <summary>
```

```
/// Display EC2 instance information.
/// </summary>
/// <param name="instance">The instance Id of the EC2 instance.</param>
public void DisplayInstanceInformation(Instance instance)
{
    Console.WriteLine($"ID: {instance.InstanceId}");
    Console.WriteLine($"Image ID: {instance.ImageId}");
    Console.WriteLine($"{{instance.InstanceType}}");
    Console.WriteLine($"Key Name: {instance.KeyName}");
    Console.WriteLine($"VPC ID: {instance.VpcId}");
    Console.WriteLine($"Public IP: {instance.PublicIpAddress}");
    Console.WriteLine($"State: {instance.State.Name}");
}

/// <summary>
/// Get information about EC2 instances with a particular state.
/// </summary>
/// <param name="tagName">The name of the tag to filter on.</param>
/// <param name="tagValue">The value of the tag to look for.</param>
/// <returns>True if successful.</returns>
public async Task<bool> GetInstancesWithState(string state)
{
    try
    {
        // Filters the results of the instance list.
        var filters = new List<Filter>
        {
            new Filter
            {
                Name = $"instance-state-name",
                Values = new List<string> { state, },
            },
        };
        var request = new DescribeInstancesRequest { Filters = filters, };

        Console.WriteLine($"\\nShowing instances with state {state}");
        var paginator = _amazonEC2.Paginators.DescribeInstances(request);

        await foreach (var response in paginator.Responses)
        {
            foreach (var reservation in response.Reservations)
            {
                foreach (var instance in reservation.Instances)
                {
```

```
        Console.WriteLine($"Instance ID: {instance.InstanceId} ");
        Console.WriteLine($"\\tCurrent State:
{instance.State.Name}");
    }
}

return true;
}
catch (AmazonEC2Exception ec2Exception)
{
    if (ec2Exception.ErrorCode == "InvalidParameterValue")
    {
        _logger.LogError(
            $"Invalid parameter value for filtering instances.");
    }

    return false;
}
catch (Exception ex)
{
    Console.WriteLine($"Couldn't list instances because: {ex.Message}");
    return false;
}
}

/// <summary>
/// Describe the instance types available.
/// </summary>
/// <returns>A list of instance type information.</returns>
public async Task<List<InstanceTypeInfo>>
DescribeInstanceTypes(ArchitectureValues architecture)
{
    try
    {
        var request = new DescribeInstanceTypesRequest();

        var filters = new List<Filter>
        {
            new Filter("processor-info.supported-architecture",
                new List<string> { architecture.ToString() })
        };
        filters.Add(new Filter("instance-type", new() { "*.micro",
            "*.small" }));
    }
}
```

```
        request.Filters = filters;
        var instanceTypes = new List<InstanceTypeInfo>();

        var paginator = _amazonEC2.Paginators.DescribeInstanceTypes(request);
        await foreach (var instanceType in paginator.InstanceTypes)
        {
            instanceTypes.Add(instanceType);
        }

        return instanceTypes;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidParameterValue")
        {
            _logger.LogError(
                $"Parameters are invalid. Ensure architecture and size
strings conform to DescribeInstanceTypes API reference.");
        }

        throw;
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
        throw;
    }
}

/// <summary>
/// Get information about an Amazon EC2 key pair.
/// </summary>
/// <param name="keyPairName">The name of the key pair.</param>
/// <returns>A list of key pair information.</returns>
public async Task<List<KeyPairInfo>> DescribeKeyPairs(string keyPairName)
{
    try
    {
        var request = new DescribeKeyPairsRequest();
        if (!string.IsNullOrEmpty(keyPairName))
        {
            request = new DescribeKeyPairsRequest

```



```
        {
            KeyNames = new List<string> { keyPairName }
        };
    }

    var response = await _amazonEC2.DescribeKeyPairsAsync(request);
    return response.KeyPairs.ToList();
}
catch (AmazonEC2Exception ec2Exception)
{
    if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
    {
        _logger.LogError(
            $"A key pair called {keyPairName} does not exist.");
    }

    throw;
}
catch (Exception ex)
{
    _logger.LogError(
        {ex.Message});
    throw;
}
}

/// <summary>
/// Retrieve information for one or all Amazon EC2 security group.
/// </summary>
/// <param name="groupId">The optional Id of a specific Amazon EC2 security
group.</param>
/// <returns>A list of security group information.</returns>
public async Task<List<SecurityGroup>> DescribeSecurityGroups(string groupId)
{
    try
    {
        var securityGroups = new List<SecurityGroup>();
        var request = new DescribeSecurityGroupsRequest();

        if (!string.IsNullOrEmpty(groupId))
        {
            var groupIds = new List<string> { groupId };

```

```
        request.GroupIds = groupIds;
    }

    var paginatorForSecurityGroups =
        _amazonEC2.Paginators.DescribeSecurityGroups(request);

    await foreach (var securityGroup in
paginatorForSecurityGroups.SecurityGroups)
    {
        securityGroups.Add(securityGroup);
    }

    return securityGroups;

}
catch (AmazonEC2Exception ec2Exception)
{
    if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
    {
        _logger.LogError(
            $"A security group {groupId} does not exist.");
    }

    throw;
}
catch (Exception ex)
{
    _logger.LogError(
        $"An error occurred while listing security groups.
{ex.Message}");
    throw;
}
}

/// <summary>
/// Display the information returned by the call to
/// DescribeSecurityGroupsAsync.
/// </summary>
/// <param name="securityGroup">A list of security group information.</param>
public void DisplaySecurityGroupInfoAsync(SecurityGroup securityGroup)
{
    Console.WriteLine($"{securityGroup.GroupName}");
    Console.WriteLine("Ingress permissions:");
    securityGroup.IpPermissions.ForEach(permission =>
```

```
{
    Console.WriteLine($"\\tFromPort: {permission.FromPort}");
    Console.WriteLine($"\\tIpProtocol: {permission.IpProtocol}");

    Console.Write($"\\tIpv4Ranges: ");
    permission.Ipv4Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIp} "); });

    Console.WriteLine($"\\n\\tIpv6Ranges:");
    permission.Ipv6Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIpv6} "); });

    Console.Write($"\\n\\tPrefixListIds: ");
    permission.PrefixListIds.ForEach(id => Console.Write($"{id.Id} "));

    Console.WriteLine($"\\n\\tTo Port: {permission.ToPort}");
});
Console.WriteLine("Egress permissions:");
securityGroup.IpPermissionsEgress.ForEach(permission =>
{
    Console.WriteLine($"\\tFromPort: {permission.FromPort}");
    Console.WriteLine($"\\tIpProtocol: {permission.IpProtocol}");

    Console.Write($"\\tIpv4Ranges: ");
    permission.Ipv4Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIp} "); });

    Console.WriteLine($"\\n\\tIpv6Ranges:");
    permission.Ipv6Ranges.ForEach(range =>
{ Console.Write($"{range.CidrIpv6} "); });

    Console.Write($"\\n\\tPrefixListIds: ");
    permission.PrefixListIds.ForEach(id => Console.Write($"{id.Id} "));

    Console.WriteLine($"\\n\\tTo Port: {permission.ToPort}");
});
}

/// <summary>
/// Disassociate an Elastic IP address from an EC2 instance.
/// </summary>
/// <param name="associationId">The association Id.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
```

```
public async Task<bool> DisassociateIp(string associationId)
{
    try
    {
        var response = await _amazonEC2.DisassociateAddressAsync(
            new DisassociateAddressRequest { AssociationId =
associationId });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidAssociationID.NotFound")
        {
            _logger.LogError(
                $"AssociationId is invalid, unable to disassociate address.
{ec2Exception.Message}");
        }

        return false;
    }
    catch (Exception ex)
    {
        _logger.LogError(
            $"An error occurred while disassociating the Elastic IP.:
{ex.Message}");
        return false;
    }
}

/// <summary>
/// Retrieve a list of available Amazon Linux images.
/// </summary>
/// <returns>A list of image information.</returns>
public async Task<List<Image>> GetEC2AmiList()
{
    var filter = new Filter { Name = "architecture", Values = new
List<string> { "x86_64" } };
    var filters = new List<Filter> { filter };
    var response = await _amazonEC2.DescribeImagesAsync(new
DescribeImagesRequest { Filters = filters });
    return response.Images;
}

/// <summary>
```

```
/// Reboot a specific EC2 instance.
/// </summary>
/// <param name="ec2InstanceId">The instance Id of the instance that will be
rebooted.</param>
/// <returns>Async Task.</returns>
public async Task<bool> RebootInstances(string ec2InstanceId)
{
    try
    {
        var request = new RebootInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId },
        };

        await _amazonEC2.RebootInstancesAsync(request);

        // Wait for the instance to be running.
        Console.WriteLine("Waiting for the instance to start.");
        await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);

        return true;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidInstanceId")
        {
            _logger.LogError(
                $"InstanceId {ec2InstanceId} is invalid, unable to reboot.
{ec2Exception.Message}");
        }
        return false;
    }
    catch (Exception ex)
    {
        _logger.LogError(
            $"An error occurred while rebooting the instance
{ec2InstanceId}.: {ex.Message}");
        return false;
    }
}

/// <summary>
/// Release an Elastic IP address. After the Elastic IP address is released,
/// it can no longer be used.
```

```
    /// </summary>
    /// <param name="allocationId">The allocation Id of the Elastic IP address.</
param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ReleaseAddress(string allocationId)
    {
        try
        {
            var request = new ReleaseAddressRequest { AllocationId =
allocationId };

            var response = await _amazonEC2.ReleaseAddressAsync(request);
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidAllocationID.NotFound")
            {
                _logger.LogError(
                    $"AllocationId {allocationId} was not found.
{ec2Exception.Message}");
            }

            return false;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while releasing the AllocationId
{allocationId}.: {ex.Message}");
            return false;
        }
    }

    /// <summary>
    /// Create and run an EC2 instance.
    /// </summary>
    /// <param name="ImageId">The image Id of the image used as a basis for the
    /// EC2 instance.</param>
    /// <param name="instanceType">The instance type of the EC2 instance to
    create.</param>
    /// <param name="keyName">The name of the key pair to associate with the
    /// instance.</param>
```

```
    /// <param name="groupId">The Id of the Amazon EC2 security group that will
    be
    /// allowed to interact with the new EC2 instance.</param>
    /// <returns>The instance Id of the new EC2 instance.</returns>
    public async Task<string> RunInstances(string imageId, string instanceType,
    string keyName, string groupId)
    {
        try
        {
            var request = new RunInstancesRequest
            {
                ImageId = imageId,
                InstanceType = instanceType,
                KeyName = keyName,
                MinCount = 1,
                MaxCount = 1,
                SecurityGroupIds = new List<string> { groupId }
            };
            var response = await _amazonEC2.RunInstancesAsync(request);
            var instanceId = response.Reservation.Instances[0].InstanceId;

            Console.WriteLine("Waiting for the instance to start.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);

            return instanceId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroupId.NotFound")
            {
                _logger.LogError(
                    $"GroupId {groupId} was not found. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while running the instance.: {ex.Message}");
            throw;
        }
    }
}
```

```
/// <summary>
/// Start an EC2 instance.
/// </summary>
/// <param name="ec2InstanceId">The instance Id of the Amazon EC2 instance
/// to start.</param>
/// <returns>Async task.</returns>
public async Task StartInstances(string ec2InstanceId)
{
    try
    {
        var request = new StartInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId },
        };

        await _amazonEC2.StartInstancesAsync(request);

        Console.WriteLine("Waiting for instance to start. ");
        await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidInstanceId")
        {
            _logger.LogError(
                $"InstanceId is invalid, unable to start.
{ec2Exception.Message}");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(
            $"An error occurred while starting the instance.: {ex.Message}");
        throw;
    }
}

/// <summary>
/// Stop an EC2 instance.
/// </summary>
/// <param name="ec2InstanceId">The instance Id of the EC2 instance to
```



```
/// stop.</param>
/// <returns>Async task.</returns>
public async Task StopInstances(string ec2InstanceId)
{
    try
    {
        var request = new StopInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId },
        };

        await _amazonEC2.StopInstancesAsync(request);
        Console.WriteLine("Waiting for the instance to stop.");
        await WaitForInstanceState(ec2InstanceId, InstanceStateName.Stopped);

        Console.WriteLine("\nThe instance has stopped.");
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidInstanceId")
        {
            _logger.LogError(
                $"InstanceId is invalid, unable to stop.
{ec2Exception.Message}");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(
            $"An error occurred while stopping the instance.: {ex.Message}");
        throw;
    }
}

/// <summary>
/// Terminate an EC2 instance.
/// </summary>
/// <param name="ec2InstanceId">The instance Id of the EC2 instance
/// to terminate.</param>
/// <returns>Async task.</returns>
public async Task<List<InstanceStateChange>> TerminateInstances(string
ec2InstanceId)
```

```
{
    try
    {
        var request = new TerminateInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId }
        };

        var response = await _amazonEC2.TerminateInstancesAsync(request);
        Console.WriteLine("Waiting for the instance to terminate.");
        await WaitForInstanceState(ec2InstanceId,
InstanceStateName.Terminated);

        Console.WriteLine($"\\nThe instance {ec2InstanceId} has been
terminated.");
        return response.TerminatingInstances;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidInstanceId")
        {
            _logger.LogError(
                $"InstanceId is invalid, unable to terminate.
{ec2Exception.Message}");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(
            $"An error occurred while terminating the instance.:
{ex.Message}");
        throw;
    }
}

/// <summary>
/// Wait until an EC2 instance is in a specified state.
/// </summary>
/// <param name="instanceId">The instance Id.</param>
/// <param name="stateName">The state to wait for.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
```

```
public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
{
    var request = new DescribeInstancesRequest
    {
        InstanceIds = new List<string> { instanceId }
    };

    // Wait until the instance is in the specified state.
    var hasState = false;
    do
    {
        // Wait 5 seconds.
        Thread.Sleep(5000);

        // Check for the desired state.
        var response = await _amazonEC2.DescribeInstancesAsync(request);
        var instance = response.Reservations[0].Instances[0];
        hasState = instance.State.Name == stateName;
        Console.WriteLine(". ");
    } while (!hasState);

    return hasState;
}
}
```

- APIEinzelheiten finden Sie unter den folgenden Themen im AWS SDK for .NET APIReferenzhandbuch.
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)
 - [DeleteSecurityGroup](#)
 - [DescribeImages](#)
 - [DescribeInstanceTypes](#)

- [DescribeInstances](#)
- [DescribeKeyPairs](#)
- [DescribeSecurityGroups](#)
- [DisassociateAddress](#)
- [ReleaseAddress](#)
- [RunInstances](#)
- [StartInstances](#)
- [StopInstances](#)
- [TerminateInstances](#)
- [UnmonitorInstances](#)

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
#####  
# function get_started_with_ec2_instances  
#  
# Runs an interactive scenario that shows how to get started using EC2 instances.  
#  
# "EC2 access" permissions are needed to run this code.  
#  
# Returns:  
# 0 - If successful.  
# 1 - If an error occurred.  
#####  
function get_started_with_ec2_instances() {  
    # Requires version 4 for mapfile.  
    local required_version=4.0
```

```
# Get the current Bash version
# Check if BASH_VERSION is set
local current_version
if [[ -n "$BASH_VERSION" ]]; then
    # Convert BASH_VERSION to a number for comparison
    current_version=$BASH_VERSION
else
    # Get the current Bash version using the bash command
    current_version=$(bash --version | head -n 1 | awk '{ print $4 }')
fi

# Convert version strings to numbers for comparison
local required_version_num current_version_num
required_version_num=$(echo "$required_version" | awk -F. '{ print ($1 * 10000)
+ ($2 * 100) + $3 }')
current_version_num=$(echo "$current_version" | awk -F. '{ print ($1 * 10000) +
($2 * 100) + $3 }')

# Compare versions
if ((current_version_num < required_version_num)); then
    echo "Error: This script requires Bash version $required_version or higher."
    echo "Your current Bash version is number is $current_version."
    exit 1
fi

{
    if [ "$EC2_OPERATIONS_SOURCED" != "True" ]; then

        source ./ec2_operations.sh
    fi
}

echo_repeat "*" 88
echo "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started with
instances demo."
echo_repeat "*" 88
echo

echo "Let's create an RSA key pair that you can be use to securely connect to "
echo "your EC2 instance."

echo -n "Enter a unique name for your key: "
get_input
local key_name
```

```
key_name=$get_input_result

local temp_dir
temp_dir=$(mktemp -d)
local key_file_name="$temp_dir/${key_name}.pem"

if ec2_create_keypair -n "${key_name}" -f "${key_file_name}"; then
    echo "Created a key pair $key_name and saved the private key to
$key_file_name"
    echo
else
    errecho "The key pair failed to create. This demo will exit."
    return 1
fi

chmod 400 "${key_file_name}"

if yes_no_input "Do you want to list some of your key pairs? (y/n) "; then
    local keys_and_fingerprints
    keys_and_fingerprints="$(ec2_describe_key_pairs)" && {
        local image_name_and_id
        while IFS=$'\n' read -r image_name_and_id; do
            local entries
            IFS=$'\t' read -ra entries <<<"$image_name_and_id"
            echo "Found rsa key ${entries[0]} with fingerprint:"
            echo "    ${entries[1]}"
        done <<<"$keys_and_fingerprints"
    }
fi

echo_repeat "*" 88
echo_repeat "*" 88

echo "Let's create a security group to manage access to your instance."
echo -n "Enter a unique name for your security group: "
get_input
local security_group_name
security_group_name=$get_input_result
local security_group_id
security_group_id=$(ec2_create_security_group -n "$security_group_name" \
-d "Security group for EC2 instance") || {
    errecho "The security failed to create. This demo will exit."
    clean_up "$key_name" "$key_file_name"
```

```
    return 1
}

echo "Security group created with ID $security_group_id"
echo

local public_ip
public_ip=$(curl -s http://checkip.amazonaws.com)

echo "Let's add a rule to allow SSH only from your current IP address."
echo "Your public IP address is $public_ip"
echo -n "press return to add this rule to your security group."
get_input

if ! ec2_authorize_security_group_ingress -g "$security_group_id" -i
"$public_ip" -p tcp -f 22 -t 22; then
    errecho "The security group rules failed to update. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
fi

echo "Security group rules updated"

local security_group_description
security_group_description="$(ec2_describe_security_groups -g
"${security_group_id}")" || {
    errecho "Failed to describe security groups. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
}

mapfile -t parameters <<<"$security_group_description"
IFS=$'\t' read -ra entries <<<"${parameters[0]}"
echo "Security group: ${entries[0]}"
echo "    ID: ${entries[1]}"
echo "    VPC: ${entries[2]}"
echo "Inbound permissions:"
IFS=$'\t' read -ra entries <<<"${parameters[1]}"
echo "    IpProtocol: ${entries[0]}"
echo "    FromPort: ${entries[1]}"
echo "    ToPort: ${entries[2]}"
echo "    CidrIp: ${parameters[2]}"

local parameters
```

```
parameters="$(ssm_get_parameters_by_path -p "/aws/service/ami-amazon-linux-
latest")" || {
    errecho "Failed to get parameters. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
}

local image_ids=""
mapfile -t parameters <<<"$parameters"
for image_name_and_id in "${parameters[@]}"; do
    IFS=$'\t' read -ra values <<<"$image_name_and_id"
    if [[ "${values[0]}" == *"amzn2"* ]]; then
        image_ids+="${values[1]} "
    fi
done

local images
images="$(ec2_describe_images -i "$image_ids")" || {
    errecho "Failed to describe images. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
}

new_line_and_tab_to_list "$images"
local images=("${list_result[@]}")

# Get the size of the array
local images_count=${#images[@]}

if ((images_count == 0)); then
    errecho "No images found. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
fi

echo_repeat "*" 88
echo_repeat "*" 88

echo "Let's create an instance from an Amazon Linux 2 AMI. Here are some
options:"
for ((i = 0; i < images_count; i += 3)); do
    echo "$(((i / 3) + 1)) - ${images[$i]}"
done
```



```

done

integer_input "Please enter the number of the AMI you want to use: " 1
"$((images_count / 3))"
local choice=$get_input_result
choice=$((choice - 1) * 3)

echo "Great choice."
echo

local architecture=${images[$((choice + 1))]}
local image_id=${images[$((choice + 2))]}
echo "Here are some instance types that support the ${architecture}
architecture of the image:"
response="$(ec2_describe_instance_types -a "${architecture}" -t
"*micro,*small")" || {
    errecho "Failed to describe instance types. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
}

local instance_types
mapfile -t instance_types <<<"$response"

# Get the size of the array
local instance_types_count=${#instance_types[@]}

echo "Here are some options:"
for ((i = 0; i < instance_types_count; i++)); do
    echo "$((i + 1)) - ${instance_types[$i]}"
done

integer_input "Which one do you want to use? " 1 "${#instance_types[@]}"
"
choice=$get_input_result
local instance_type=${instance_types[$((choice - 1))]}
echo "Another great choice."
echo

echo "Creating your instance and waiting for it to start..."
local instance_id
instance_id=$(ec2_run_instances -i "$image_id" -t "$instance_type" -k
"$key_name" -s "$security_group_id") || {
    errecho "Failed to run instance. This demo will exit."
}

```

```
    clean_up "$key_name" "$key_file_name" "$security_group_id"
    return 1
}

ec2_wait_for_instance_running -i "$instance_id"
echo "Your instance is ready:"
echo

local instance_details
instance_details="$(ec2_describe_instances -i "${instance_id}")"

echo
print_instance_details "${instance_details}"

local public_ip
public_ip=$(echo "${instance_details}" | awk '{print $6}')
echo
echo "You can use SSH to connect to your instance"
echo "If the connection attempt times out, you might have to manually update
the SSH ingress rule"
echo "for your IP address in the AWS Management Console."
connect_to_instance "$key_file_name" "$public_ip"

echo -n "Press Enter when you're ready to continue the demo: "
get_input

echo_repeat "*" 88
echo_repeat "*" 88

echo "Let's stop and start your instance to see what changes."
echo "Stopping your instance and waiting until it's stopped..."
ec2_stop_instances -i "$instance_id"
ec2_wait_for_instance_stopped -i "$instance_id"

echo "Your instance is stopped. Restarting..."

ec2_start_instances -i "$instance_id"
ec2_wait_for_instance_running -i "$instance_id"

echo "Your instance is running again."
local instance_details
instance_details="$(ec2_describe_instances -i "${instance_id}")"

print_instance_details "${instance_details}"
```

```
public_ip=$(echo "${instance_details}" | awk '{print $6}')

echo "Every time your instance is restarted, its public IP address changes"
connect_to_instance "$key_file_name" "$public_ip"

echo -n "Press Enter when you're ready to continue the demo: "
get_input

echo_repeat "*" 88
echo_repeat "*" 88

echo "You can allocate an Elastic IP address and associate it with your
instance"
echo "to keep a consistent IP address even when your instance restarts."

local result
result=$(ec2_allocate_address -d vpc) || {
    errecho "Failed to allocate an address. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id" "$instance_id"
    return 1
}

local elastic_ip allocation_id
elastic_ip=$(echo "$result" | awk '{print $1}')
allocation_id=$(echo "$result" | awk '{print $2}')

echo "Allocated static Elastic IP address: $elastic_ip"

local association_id
association_id=$(ec2_associate_address -i "$instance_id" -a "$allocation_id")
|| {
    errecho "Failed to associate an address. This demo will exit."
    clean_up "$key_name" "$key_file_name" "$security_group_id" "$instance_id"
"$allocation_id"
    return 1
}

echo "Associated your Elastic IP with your instance."
echo "You can now use SSH to connect to your instance by using the Elastic IP."
connect_to_instance "$key_file_name" "$elastic_ip"

echo -n "Press Enter when you're ready to continue the demo: "
get_input
```

```
echo_repeat "*" 88
echo_repeat "*" 88

echo "Let's stop and start your instance to see what changes."
echo "Stopping your instance and waiting until it's stopped..."
ec2_stop_instances -i "$instance_id"
ec2_wait_for_instance_stopped -i "$instance_id"

echo "Your instance is stopped. Restarting..."

ec2_start_instances -i "$instance_id"
ec2_wait_for_instance_running -i "$instance_id"

echo "Your instance is running again."
local instance_details
instance_details="$(ec2_describe_instances -i "${instance_id}")"

print_instance_details "${instance_details}"

echo "Because you have associated an Elastic IP with your instance, you can"
echo "connect by using a consistent IP address after the instance restarts."
connect_to_instance "$key_file_name" "$elastic_ip"

echo -n "Press Enter when you're ready to continue the demo: "
get_input

echo_repeat "*" 88
echo_repeat "*" 88

if yes_no_input "Do you want to delete the resources created in this demo: (y/
n) "; then
    clean_up "$key_name" "$key_file_name" "$security_group_id" "$instance_id" \
        "$allocation_id" "$association_id"
else
    echo "The following resources were not deleted."
    echo "Key pair: $key_name"
    echo "Key file: $key_file_name"
    echo "Security group: $security_group_id"
    echo "Instance: $instance_id"
    echo "Elastic IP address: $elastic_ip"
fi
}
```

```
#####  
# function clean_up  
#  
# This function cleans up the created resources.  
#   $1 - The name of the ec2 key pair to delete.  
#   $2 - The name of the key file to delete.  
#   $3 - The ID of the security group to delete.  
#   $4 - The ID of the instance to terminate.  
#   $5 - The ID of the elastic IP address to release.  
#   $6 - The ID of the elastic IP address to disassociate.  
#  
# Returns:  
#   0 - If successful.  
#   1 - If an error occurred.  
#####  
function clean_up() {  
    local result=0  
    local key_pair_name=$1  
    local key_file_name=$2  
    local security_group_id=$3  
    local instance_id=$4  
    local allocation_id=$5  
    local association_id=$6  
  
    if [ -n "$association_id" ]; then  
        # bashsupport disable=BP2002  
        if (ec2_disassociate_address -a "$association_id"); then  
            echo "Disassociated elastic IP address with ID $association_id"  
        else  
            errecho "The elastic IP address disassociation failed."  
            result=1  
        fi  
    fi  
  
    if [ -n "$allocation_id" ]; then  
        # bashsupport disable=BP2002  
        if (ec2_release_address -a "$allocation_id"); then  
            echo "Released elastic IP address with ID $allocation_id"  
        else  
            errecho "The elastic IP address release failed."  
            result=1  
        fi  
    fi  
fi
```

```
if [ -n "$instance_id" ]; then
    # bashsupport disable=BP2002
    if (ec2_terminate_instances -i "$instance_id"); then
        echo "Started terminating instance with ID $instance_id"

        ec2_wait_for_instance_terminated -i "$instance_id"
    else
        errecho "The instance terminate failed."
        result=1
    fi
fi

if [ -n "$security_group_id" ]; then
    # bashsupport disable=BP2002
    if (ec2_delete_security_group -i "$security_group_id"); then
        echo "Deleted security group with ID $security_group_id"
    else
        errecho "The security group delete failed."
        result=1
    fi
fi

if [ -n "$key_pair_name" ]; then
    # bashsupport disable=BP2002
    if (ec2_delete_keypair -n "$key_pair_name"); then
        echo "Deleted key pair named $key_pair_name"
    else
        errecho "The key pair delete failed."
        result=1
    fi
fi

if [ -n "$key_file_name" ]; then
    rm -f "$key_file_name"
fi

return $result
}

#####
# function ssm_get_parameters_by_path
#
# This function retrieves one or more parameters from the AWS Systems Manager
Parameter Store
```

```

# by specifying a parameter path.
#
# Parameters:
#     -p parameter_path - The path of the parameter(s) to retrieve.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function ssm_get_parameters_by_path() {
    local parameter_path response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ssm_get_parameters_by_path"
        echo "Retrieves one or more parameters from the AWS Systems Manager Parameter
Store by specifying a parameter path."
        echo "  -p parameter_path - The path of the parameter(s) to retrieve."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "p:h" option; do
        case "${option}" in
            p) parameter_path="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$parameter_path" ]]; then
        errecho "ERROR: You must provide a parameter path with the -p parameter."
        usage
        return 1
    fi
}

```

```

response=$(aws ssm get-parameters-by-path \
  --path "$parameter_path" \
  --query "Parameters[*].[Name, Value]" \
  --output text) || {
  aws_cli_error_log $?
  errecho "ERROR: AWS reports get-parameters-by-path operation failed.
$response"
  return 1
}

echo "$response"

return 0
}

#####
# function print_instance_details
#
# This function prints the details of an Amazon Elastic Compute Cloud (Amazon
# EC2) instance.
#
# Parameters:
#   instance_details - The instance details in the format "InstanceId ImageId
#   InstanceType KeyName VpcId PublicIpAddress State.Name".
#
# Returns:
#   0 - If successful.
#   1 - If it fails.
#####
function print_instance_details() {
  local instance_details="$1"

  if [[ -z "${instance_details}" ]]; then
    echo "Error: Missing required instance details argument."
    return 1
  fi

  local instance_id image_id instance_type key_name vpc_id public_ip state
  instance_id=$(echo "${instance_details}" | awk '{print $1}')
  image_id=$(echo "${instance_details}" | awk '{print $2}')
  instance_type=$(echo "${instance_details}" | awk '{print $3}')
  key_name=$(echo "${instance_details}" | awk '{print $4}')
  vpc_id=$(echo "${instance_details}" | awk '{print $5}')

```



```

public_ip=$(echo "${instance_details}" | awk '{print $6}')
state=$(echo "${instance_details}" | awk '{print $7}')

echo "    ID: ${instance_id}"
echo "    Image ID: ${image_id}"
echo "    Instance type: ${instance_type}"
echo "    Key name: ${key_name}"
echo "    VPC ID: ${vpc_id}"
echo "    Public IP: ${public_ip}"
echo "    State: ${state}"

return 0
}

#####
# function connect_to_instance
#
# This function displays the public IP address of an Amazon Elastic Compute Cloud
# (Amazon EC2) instance and prompts the user to connect to the instance via SSH.
#
# Parameters:
#     $1 - The name of the key file used to connect to the instance.
#     $2 - The public IP address of the instance.
#
# Returns:
#     None
#####
function connect_to_instance() {
    local key_file_name="$1"
    local public_ip="$2"

    # Validate the input parameters
    if [[ -z "$key_file_name" ]]; then
        echo "ERROR: You must provide a key file name as the first argument." >&2
        return 1
    fi

    if [[ -z "$public_ip" ]]; then
        echo "ERROR: You must provide a public IP address as the second argument."
    >&2
        return 1
    fi

    # Display the public IP address and connection command

```

```

echo "To connect, run the following command:"
echo "    ssh -i ${key_file_name} ec2-user@${public_ip}"

# Prompt the user to connect to the instance
if yes_no_input "Do you want to connect now? (y/n) "; then
    echo "After you have connected, you can return to this example by typing
'exit'"
    ssh -i "${key_file_name}" ec2-user@"${public_ip}"
fi
}

#####
# function get_input
#
# This function gets user input from the command line.
#
# Outputs:
#   User input to stdout.
#
# Returns:
#   0
#####
function get_input() {

    if [ -z "${mock_input+x}" ]; then
        read -r get_input_result
    else

        if [ "$mock_input_array_index" -lt ${#mock_input_array[@]} ]; then
            get_input_result="${mock_input_array[$mock_input_array_index]}"
            # bashsupport disable=BP2001
            # shellcheck disable=SC2206
            ((mock_input_array_index++))
            echo -n "$get_input_result"
        else
            echo "MOCK_INPUT_ARRAY has no more elements" 1>&2
            return 1
        fi
    fi

    return 0
}

#####

```

```
# function yes_no_input
#
# This function requests a yes/no answer from the user, following to a prompt.
#
# Parameters:
#     $1 - The prompt.
#
# Returns:
#     0 - If yes.
#     1 - If no.
#####
function yes_no_input() {
    if [ -z "$1" ]; then
        echo "Internal error yes_no_input"
        return 1
    fi

    local index=0
    local response="N"
    while [[ $index -lt 10 ]]; do
        index=$((index + 1))
        echo -n "$1"
        if ! get_input; then
            return 1
        fi
        response=$(echo "$get_input_result" | tr '[:upper:]' '[:lower:]')
        if [ "$response" = "y" ] || [ "$response" = "n" ]; then
            break
        else
            echo -e "\nPlease enter or 'y' or 'n'."
        fi
    done

    echo

    if [ "$response" = "y" ]; then
        return 0
    else
        return 1
    fi
}
#####
# function integer_input
```

```
#
# This function prompts the user to enter an integer within a specified range
# and validates the input.
#
# Parameters:
#     $1 - The prompt message to display to the user.
#     $2 - The minimum value of the accepted range.
#     $3 - The maximum value of the accepted range.
#
# Returns:
#     The valid integer input from the user.
#     If the input is invalid or out of range, the function will continue
#     prompting the user until a valid input is provided.
#####
function integer_input() {
    local prompt="$1"
    local min_value="$2"
    local max_value="$3"
    local input=""

    while true; do
        # Display the prompt message and wait for user input
        echo -n "$prompt"

        if ! get_input; then
            return 1
        fi

        input="$get_input_result"

        # Check if the input is a valid integer
        if [[ "$input" =~ ^-?[0-9]+$ ]]; then
            # Check if the input is within the specified range
            if ((input >= min_value && input <= max_value)); then
                return 0
            else
                echo "Error: Input, $input, must be between $min_value and $max_value."
            fi
        else
            echo "Error: Invalid input- $input. Please enter an integer."
        fi
    done
}
#####
```

```
# function new_line_and_tab_to_list
#
# This function takes a string input containing newlines and tabs, and
# converts it into a list (array) of elements.
#
# Parameters:
#     $1 - The input string containing newlines and tabs.
#
# Returns:
#     The resulting list (array) is stored in the global variable
#     'list_result'.
#####
function new_line_and_tab_to_list() {
    local input=$1
    export list_result

    list_result=()
    mapfile -t lines <<<"$input"
    local line
    for line in "${lines[@]"; do
        IFS=$'\t' read -ra parameters <<<"$line"
        list_result+=("${parameters[@]}")
    done
}

#####
# function echo_repeat
#
# This function prints a string 'n' times to stdout.
#
# Parameters:
#     $1 - The string.
#     $2 - Number of times to print the string.
#
# Outputs:
#     String 'n' times to stdout.
#
# Returns:
#     0
#####
function echo_repeat() {
    local end=$2
    for ((i = 0; i < end; i++)); do
        echo -n "$1"
    done
}
```

```
done
echo
}
```

Die in diesem Szenario verwendeten „DynamoDB“-Funktionen.

```
#####
# function ec2_create_keypair
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or
# 2048-bit RSA key pair
# and writes it to a file.
#
# Parameters:
#     -n key_pair_name - A key pair name.
#     -f file_path - File to store the key pair.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_create_keypair() {
    local key_pair_name file_path response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_create_keypair"
        echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or 2048-
bit RSA key pair"
        echo " and writes it to a file."
        echo "  -n key_pair_name - A key pair name."
        echo "  -f file_path - File to store the key pair."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:f:h" option; do
        case "${option}" in
            n) key_pair_name="${OPTARG}" ;;
            f) file_path="${OPTARG}" ;;
            h)

```

```

        usage
        return 0
        ;;
    \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
done
export OPTIND=1

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

if [[ -z "$file_path" ]]; then
    errecho "ERROR: You must provide a file path with the -f parameter."
    usage
    return 1
fi

response=$(aws ec2 create-key-pair \
    --key-name "$key_pair_name" \
    --query 'KeyMaterial' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
}

if [[ -n "$file_path" ]]; then
    echo "$response" >"$file_path"
fi

return 0
}

#####
# function ec2_describe_key_pairs
#

```

```

# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
  key pairs.
#
# Parameters:
#     -h - Display help.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_describe_key_pairs() {
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_describe_key_pairs"
        echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2) key
pairs."
        echo "  -h - Display help."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "h" option; do
        case "${option}" in
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    local response

    response=$(aws ec2 describe-key-pairs \
        --query 'KeyPairs[*].[KeyName, KeyFingerprint]' \
        --output text) || {
        aws_cli_error_log ${?}
    }
}

```



```

    errecho "ERROR: AWS reports describe-key-pairs operation failed.$response"
    return 1
}

echo "$response"

return 0
}

#####
# function ec2_create_security_group
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) security
# group.
#
# Parameters:
#     -n security_group_name - The name of the security group.
#     -d security_group_description - The description of the security group.
#
# Returns:
#     The ID of the created security group, or an error message if the
#     operation fails.
# And:
#     0 - If successful.
#     1 - If it fails.
#
#####
function ec2_create_security_group() {
    local security_group_name security_group_description response

    # Function to display usage information
    function usage() {
        echo "function ec2_create_security_group"
        echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group."
        echo "  -n security_group_name - The name of the security group."
        echo "  -d security_group_description - The description of the security
group."
        echo ""
    }

    # Parse the command-line arguments
    while getopts "n:d:h" option; do
        case "${option}" in
            n) security_group_name="${OPTARG}" ;;

```

```

    d) security_group_description="${OPTARG}" ;;
  h)
    usage
    return 0
    ;;
  \?)
    echo "Invalid parameter"
    usage
    return 1
    ;;
esac
done
export OPTIND=1

# Validate the input parameters
if [[ -z "$security_group_name" ]]; then
  errecho "ERROR: You must provide a security group name with the -n
parameter."
  return 1
fi

if [[ -z "$security_group_description" ]]; then
  errecho "ERROR: You must provide a security group description with the -d
parameter."
  return 1
fi

# Create the security group
response=$(aws ec2 create-security-group \
  --group-name "$security_group_name" \
  --description "$security_group_description" \
  --query "GroupId" \
  --output text) || {
  aws_cli_error_log ${?}
  errecho "ERROR: AWS reports create-security-group operation failed."
  errecho "$response"
  return 1
}

echo "$response"
return 0
}

#####

```

```

# function ec2_describe_security_groups
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
# security groups.
#
# Parameters:
#     -g security_group_id - The ID of the security group to describe
#     (optional).
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_describe_security_groups() {
    local security_group_id response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_describe_security_groups"
        echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
security groups."
        echo "  -g security_group_id - The ID of the security group to describe
(optional)."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "g:h" option; do
        case "${option}" in
            g) security_group_id="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

```

```

    local query="SecurityGroups[*].[GroupName, GroupId, VpcId, IpPermissions[*].
[IpProtocol, FromPort, ToPort, IpRanges[*].CidrIp]]"

    if [[ -n "$security_group_id" ]]; then
        response=$(aws ec2 describe-security-groups --group-ids "$security_group_id"
--query "${query}" --output text)
    else
        response=$(aws ec2 describe-security-groups --query "${query}" --output text)
    fi

    local error_code=${?}

    if [[ $error_code -ne 0 ]]; then
        aws_cli_error_log $error_code
        errecho "ERROR: AWS reports describe-security-groups operation failed.
$response"
        return 1
    fi

    echo "$response"

    return 0
}

#####
# function ec2_authorize_security_group_ingress
#
# This function authorizes an ingress rule for an Amazon Elastic Compute Cloud
(Amazon EC2) security group.
#
# Parameters:
#     -g security_group_id - The ID of the security group.
#     -i ip_address - The IP address or CIDR block to authorize.
#     -p protocol - The protocol to authorize (e.g., tcp, udp, icmp).
#     -f from_port - The start of the port range to authorize.
#     -t to_port - The end of the port range to authorize.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_authorize_security_group_ingress() {
    local security_group_id ip_address protocol from_port to_port response
    local option OPTARG # Required to use getopt command in a function.

```

```
# bashsupport disable=BP5008
function usage() {
    echo "function ec2_authorize_security_group_ingress"
    echo "Authorizes an ingress rule for an Amazon Elastic Compute Cloud (Amazon
EC2) security group."
    echo "  -g security_group_id - The ID of the security group."
    echo "  -i ip_address - The IP address or CIDR block to authorize."
    echo "  -p protocol - The protocol to authorize (e.g., tcp, udp, icmp)."
    echo "  -f from_port - The start of the port range to authorize."
    echo "  -t to_port - The end of the port range to authorize."
    echo ""
}

# Retrieve the calling parameters.
while getopts "g:i:p:f:t:h" option; do
    case "${option}" in
        g) security_group_id="${OPTARG}" ;;
        i) ip_address="${OPTARG}" ;;
        p) protocol="${OPTARG}" ;;
        f) from_port="${OPTARG}" ;;
        t) to_port="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -g parameter."
    usage
    return 1
fi

if [[ -z "$ip_address" ]]; then
    errecho "ERROR: You must provide an IP address or CIDR block with the -i
parameter."
```

```
usage
return 1
fi

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

if [[ -z "$from_port" ]]; then
    errecho "ERROR: You must provide a start port with the -f parameter."
    usage
    return 1
fi

if [[ -z "$to_port" ]]; then
    errecho "ERROR: You must provide an end port with the -t parameter."
    usage
    return 1
fi

response=$(aws ec2 authorize-security-group-ingress \
    --group-id "$security_group_id" \
    --cidr "${ip_address}/32" \
    --protocol "$protocol" \
    --port "$from_port-$to_port" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports authorize-security-group-ingress operation
failed.$response"
    return 1
}

return 0
}

#####
# function ec2_describe_images
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
images.
#
# Parameters:
```

```

#     -i image_ids - A space-separated list of image IDs (optional).
#     -h - Display help.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_describe_images() {
    local image_ids response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_describe_images"
        echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
images."
        echo "  -i image_ids - A space-separated list of image IDs (optional)."
        echo "  -h - Display help."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "i:h" option; do
        case "${option}" in
            i) image_ids="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    local aws_cli_args=()

    if [[ -n "$image_ids" ]]; then
        # shellcheck disable=SC2206
        aws_cli_args+=("--image-ids" $image_ids)
    fi

```

```

response=$(aws ec2 describe-images \
  "${aws_cli_args[@]}" \
  --query 'Images[*].[Description,Architecture,ImageId]' \
  --output text) || {
  aws_cli_error_log ${?}
  errecho "ERROR: AWS reports describe-images operation failed.$response"
  return 1
}

echo "$response"

return 0
}

#####
# ec2_describe_instance_types
#
# This function describes EC2 instance types filtered by processor architecture
# and optionally by instance type. It takes the following arguments:
#
# -a, --architecture ARCHITECTURE Specify the processor architecture (e.g.,
# x86_64)
# -t, --type INSTANCE_TYPE Comma-separated list of instance types (e.g.,
# t2.micro)
# -h, --help Show the usage help
#
# The function prints the instance type and supported architecture for each
# matching instance type.
#####
function ec2_describe_instance_types() {
  local architecture=""
  local instance_types=""

  # bashsupport disable=BP5008
  function usage() {
    echo "Usage: ec2_describe_instance_types [-a|--architecture ARCHITECTURE] [-t|--type INSTANCE_TYPE] [-h|--help]"
    echo "  -a, --architecture ARCHITECTURE Specify the processor architecture (e.g., x86_64)"
    echo "  -t, --type INSTANCE_TYPE Comma-separated list of instance types (e.g., t2.micro)"
    echo "  -h, --help Show this help message"
  }
}

```



```
while [[ $# -gt 0 ]]; do
  case "$1" in
    -a | --architecture)
      architecture="$2"
      shift 2
      ;;
    -t | --type)
      instance_types="$2"
      shift 2
      ;;
    -h | --help)
      usage
      return 0
      ;;
    *)
      echo "Unknown argument: $1"
      return 1
      ;;
  esac
done

if [[ -z "$architecture" ]]; then
  errecho "Error: Architecture not specified."
  usage
  return 1
fi

if [[ -z "$instance_types" ]]; then
  errecho "Error: Instance type not specified."
  usage
  return 1
fi

local tmp_json_file="temp_ec2.json"
echo -n '[
  {
    "Name": "processor-info.supported-architecture",
    "Values": [' >"$tmp_json_file"

local items
IFS=',' read -ra items <<<"$architecture"
local array_size
array_size=${#items[@]}
```

```

for i in $(seq 0 $((array_size - 1))); do
    echo -n '""${items[$i]}""' >>"$tmp_json_file"
    if [[ $i -lt $((array_size - 1)) ]]; then
        echo -n ',' >>"$tmp_json_file"
    fi
done
echo -n ']],
{
    "Name": "instance-type",
    "Values": [' >>"$tmp_json_file"
IFS=', ' read -ra items <<<"$instance_types"
local array_size
array_size=${#items[@]}
for i in $(seq 0 $((array_size - 1))); do
    echo -n '""${items[$i]}""' >>"$tmp_json_file"
    if [[ $i -lt $((array_size - 1)) ]]; then
        echo -n ',' >>"$tmp_json_file"
    fi
done

echo -n ']]]' >>"$tmp_json_file"

local response
response=$(aws ec2 describe-instance-types --filters file://"${tmp_json_file}" \
    --query 'InstanceTypes[*].[InstanceType]' --output text)

local error_code=$?

rm "$tmp_json_file"

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    echo "ERROR: AWS reports describe-instance-types operation failed."
    return 1
fi

echo "$response"
return 0
}

#####
# function ec2_run_instances
#

```

```

# This function launches one or more Amazon Elastic Compute Cloud (Amazon EC2)
instances.
#
# Parameters:
#     -i image_id - The ID of the Amazon Machine Image (AMI) to use.
#     -t instance_type - The instance type to use (e.g., t2.micro).
#     -k key_pair_name - The name of the key pair to use.
#     -s security_group_id - The ID of the security group to use.
#     -c count - The number of instances to launch (default: 1).
#     -h - Display help.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_run_instances() {
    local image_id instance_type key_pair_name security_group_id count response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_run_instances"
        echo "Launches one or more Amazon Elastic Compute Cloud (Amazon EC2)
instances."
        echo "  -i image_id - The ID of the Amazon Machine Image (AMI) to use."
        echo "  -t instance_type - The instance type to use (e.g., t2.micro)."
        echo "  -k key_pair_name - The name of the key pair to use."
        echo "  -s security_group_id - The ID of the security group to use."
        echo "  -c count - The number of instances to launch (default: 1)."
        echo "  -h - Display help."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "i:t:k:s:c:h" option; do
        case "${option}" in
            i) image_id="${OPTARG}" ;;
            t) instance_type="${OPTARG}" ;;
            k) key_pair_name="${OPTARG}" ;;
            s) security_group_id="${OPTARG}" ;;
            c) count="${OPTARG}" ;;
            h)
                usage
                return 0
        esac
    done
}

```

```
    ;;
  \?)
    echo "Invalid parameter"
    usage
    return 1
    ;;
esac
done
export OPTIND=1

if [[ -z "$image_id" ]]; then
  errecho "ERROR: You must provide an Amazon Machine Image (AMI) ID with the -i
parameter."
  usage
  return 1
fi

if [[ -z "$instance_type" ]]; then
  errecho "ERROR: You must provide an instance type with the -t parameter."
  usage
  return 1
fi

if [[ -z "$key_pair_name" ]]; then
  errecho "ERROR: You must provide a key pair name with the -k parameter."
  usage
  return 1
fi

if [[ -z "$security_group_id" ]]; then
  errecho "ERROR: You must provide a security group ID with the -s parameter."
  usage
  return 1
fi

if [[ -z "$count" ]]; then
  count=1
fi

response=$(aws ec2 run-instances \
  --image-id "$image_id" \
  --instance-type "$instance_type" \
  --key-name "$key_pair_name" \
  --security-group-ids "$security_group_id" \
```

```

--count "$count" \
--query 'Instances[*].[InstanceId]' \
--output text) || {
aws_cli_error_log ${?}
errecho "ERROR: AWS reports run-instances operation failed.$response"
return 1
}

echo "$response"

return 0
}

#####
# function ec2_describe_instances
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
# instances.
#
# Parameters:
#     -i instance_id - The ID of the instance to describe (optional).
#     -q query - The query to filter the response (optional).
#     -h - Display help.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_describe_instances() {
    local instance_id query response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_describe_instances"
        echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
instances."
        echo "  -i instance_id - The ID of the instance to describe (optional)."
        echo "  -q query - The query to filter the response (optional)."
        echo "  -h - Display help."
        echo ""
    }

    # Retrieve the calling parameters.

```

```
while getopts "i:q:h" option; do
  case "${option}" in
    i) instance_id="${OPTARG}" ;;
    q) query="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

local aws_cli_args=()

if [[ -n "$instance_id" ]]; then
  # shellcheck disable=SC2206
  aws_cli_args+=("--instance-ids" $instance_id)
fi

local query_arg=""
if [[ -n "$query" ]]; then
  query_arg="--query '$query'"
else
  query_arg="--query Reservations[*].Instances[*].
[InstanceId,ImageId,InstanceType,KeyName,VpcId,PublicIpAddress,State.Name]"
fi

# shellcheck disable=SC2086
response=$(aws ec2 describe-instances \
  "${aws_cli_args[@]}" \
  $query_arg \
  --output text) || {
  aws_cli_error_log ${?}
  errecho "ERROR: AWS reports describe-instances operation failed.$response"
  return 1
}

echo "$response"
```

```

return 0
}

#####
# function ec2_stop_instances
#
# This function stops one or more Amazon Elastic Compute Cloud (Amazon EC2)
instances.
#
# Parameters:
#     -i instance_id - The ID(s) of the instance(s) to stop (comma-separated).
#     -h - Display help.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_stop_instances() {
    local instance_ids
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_stop_instances"
        echo "Stops one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
        echo "  -i instance_id - The ID(s) of the instance(s) to stop (comma-
separated)."
        echo "  -h - Display help."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "i:h" option; do
        case "${option}" in
            i) instance_ids="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
}

```

```

    esac
done
export OPTIND=1

if [[ -z "$instance_ids" ]]; then
    errecho "ERROR: You must provide one or more instance IDs with the -i
parameter."
    usage
    return 1
fi

response=$(aws ec2 stop-instances \
    --instance-ids "${instance_ids}") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports stop-instances operation failed with $response."
    return 1
}

return 0
}

#####
# function ec2_start_instances
#
# This function starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
instances.
#
# Parameters:
#     -i instance_id - The ID(s) of the instance(s) to start (comma-separated).
#     -h - Display help.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_start_instances() {
    local instance_ids
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_start_instances"
        echo "Starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
instances."

```



```
    echo " -i instance_id - The ID(s) of the instance(s) to start (comma-
separated)."
```

```
    echo " -h - Display help."
```

```
    echo ""
```

```
  }
```

```
  # Retrieve the calling parameters.
```

```
  while getopts "i:h" option; do
```

```
    case "${option}" in
```

```
      i) instance_ids="${OPTARG}" ;;
```

```
      h)
```

```
        usage
```

```
        return 0
```

```
        ;;
```

```
      \?)
```

```
        echo "Invalid parameter"
```

```
        usage
```

```
        return 1
```

```
        ;;
```

```
    esac
```

```
  done
```

```
  export OPTIND=1
```

```
  if [[ -z "$instance_ids" ]]; then
```

```
    errecho "ERROR: You must provide one or more instance IDs with the -i
```

```
parameter."
```

```
    usage
```

```
    return 1
```

```
  fi
```

```
  response=$(aws ec2 start-instances \
```

```
    --instance-ids "${instance_ids}") || {
```

```
    aws_cli_error_log ${?}
```

```
    errecho "ERROR: AWS reports start-instances operation failed with $response."
```

```
    return 1
```

```
  }
```

```
  return 0
```

```
}
```

```
#####
```

```
# function ec2_allocate_address
```

```
#
```

```
# This function allocates an Elastic IP address for use with Amazon Elastic
  Compute Cloud (Amazon EC2) instances in a specific AWS Region.
#
# Parameters:
#     -d domain - The domain for the Elastic IP address (either 'vpc' or
  'standard').
#
# Returns:
#     The allocated Elastic IP address, or an error message if the operation
  fails.
# And:
#     0 - If successful.
#     1 - If it fails.
#
#####
function ec2_allocate_address() {
    local domain response

    # Function to display usage information
    function usage() {
        echo "function ec2_allocate_address"
        echo "Allocates an Elastic IP address for use with Amazon Elastic Compute
  Cloud (Amazon EC2) instances in a specific AWS Region."
        echo " -d domain - The domain for the Elastic IP address (either 'vpc' or
  'standard')."
        echo ""
    }

    # Parse the command-line arguments
    while getopts "d:h" option; do
        case "${option}" in
            d) domain="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1
```

```
# Validate the input parameters
if [[ -z "$domain" ]]; then
    errecho "ERROR: You must provide a domain with the -d parameter (either 'vpc'
or 'standard')."
    return 1
fi

if [[ "$domain" != "vpc" && "$domain" != "standard" ]]; then
    errecho "ERROR: Invalid domain value. Must be either 'vpc' or 'standard'."
    return 1
fi

# Allocate the Elastic IP address
response=$(aws ec2 allocate-address \
    --domain "$domain" \
    --query "[PublicIp,AllocationId]" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports allocate-address operation failed."
    errecho "$response"
    return 1
}

echo "$response"
return 0
}

#####
# function ec2_associate_address
#
# This function associates an Elastic IP address with an Amazon Elastic Compute
Cloud (Amazon EC2) instance.
#
# Parameters:
#     -a allocation_id - The allocation ID of the Elastic IP address to
associate.
#     -i instance_id - The ID of the EC2 instance to associate the Elastic IP
address with.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#
```

```
#####  
function ec2_associate_address() {  
    local allocation_id instance_id response  
  
    # Function to display usage information  
    function usage() {  
        echo "function ec2_associate_address"  
        echo "Associates an Elastic IP address with an Amazon Elastic Compute Cloud  
(Amazon EC2) instance."  
        echo " -a allocation_id - The allocation ID of the Elastic IP address to  
associate."  
        echo " -i instance_id - The ID of the EC2 instance to associate the Elastic  
IP address with."  
        echo ""  
    }  
  
    # Parse the command-line arguments  
    while getopts "a:i:h" option; do  
        case "${option}" in  
            a) allocation_id="${OPTARG}" ;;  
            i) instance_id="${OPTARG}" ;;  
            h)  
                usage  
                return 0  
                ;;  
            \?)  
                echo "Invalid parameter"  
                usage  
                return 1  
                ;;  
        esac  
    done  
    export OPTIND=1  
  
    # Validate the input parameters  
    if [[ -z "$allocation_id" ]]; then  
        errecho "ERROR: You must provide an allocation ID with the -a parameter."  
        return 1  
    fi  
  
    if [[ -z "$instance_id" ]]; then  
        errecho "ERROR: You must provide an instance ID with the -i parameter."  
        return 1  
    fi  
}
```

```

# Associate the Elastic IP address
response=$(aws ec2 associate-address \
  --allocation-id "$allocation_id" \
  --instance-id "$instance_id" \
  --query "AssociationId" \
  --output text) || {
  aws_cli_error_log ${?}
  errecho "ERROR: AWS reports associate-address operation failed."
  errecho "$response"
  return 1
}

echo "$response"
return 0
}

#####
# function ec2_disassociate_address
#
# This function disassociates an Elastic IP address from an Amazon Elastic
# Compute Cloud (Amazon EC2) instance.
#
# Parameters:
#   -a association_id - The association ID that represents the association of
#   the Elastic IP address with an instance.
#
# And:
#   0 - If successful.
#   1 - If it fails.
#
#####
function ec2_disassociate_address() {
  local association_id response

  # Function to display usage information
  function usage() {
    echo "function ec2_disassociate_address"
    echo "Disassociates an Elastic IP address from an Amazon Elastic Compute
    Cloud (Amazon EC2) instance."
    echo "  -a association_id - The association ID that represents the
    association of the Elastic IP address with an instance."
    echo ""
  }
}

```

```

# Parse the command-line arguments
while getopts "a:h" option; do
  case "${option}" in
    a) association_id="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

# Validate the input parameters
if [[ -z "$association_id" ]]; then
  errecho "ERROR: You must provide an association ID with the -a parameter."
  return 1
fi

response=$(aws ec2 disassociate-address \
  --association-id "$association_id") || {
  aws_cli_error_log ${?}
  errecho "ERROR: AWS reports disassociate-address operation failed."
  errecho "$response"
  return 1
}

return 0
}

#####
# function ec2_release_address
#
# This function releases an Elastic IP address from an Amazon Elastic Compute
# Cloud (Amazon EC2) instance.
#
# Parameters:
#   -a allocation_id - The allocation ID of the Elastic IP address to
#   release.

```

```
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#
#####
function ec2_release_address() {
    local allocation_id response

    # Function to display usage information
    function usage() {
        echo "function ec2_release_address"
        echo "Releases an Elastic IP address from an Amazon Elastic Compute Cloud
(Amazon EC2) instance."
        echo " -a allocation_id - The allocation ID of the Elastic IP address to
release."
        echo ""
    }

    # Parse the command-line arguments
    while getopts "a:h" option; do
        case "${option}" in
            a) allocation_id="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    # Validate the input parameters
    if [[ -z "$allocation_id" ]]; then
        errecho "ERROR: You must provide an allocation ID with the -a parameter."
        return 1
    fi

    response=$(aws ec2 release-address \
        --allocation-id "$allocation_id") || {
```

```
aws_cli_error_log ${?}
errecho "ERROR: AWS reports release-address operation failed."
errecho "$response"
return 1
}

return 0
}

#####
# function ec2_terminate_instances
#
# This function terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
# instances using the AWS CLI.
#
# Parameters:
#     -i instance_ids - A space-separated list of instance IDs.
#     -h - Display help.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_terminate_instances() {
    local instance_ids response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_terminate_instances"
        echo "Terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
instances."
        echo "  -i instance_ids - A space-separated list of instance IDs."
        echo "  -h - Display help."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "i:h" option; do
        case "${option}" in
            i) instance_ids="${OPTARG}" ;;
            h)
                usage
                return 0
        esac
    done
}
```



```

        ;;
    \?)
        echo "Invalid parameter"
        usage
        return 1
    ;;
esac
done
export OPTIND=1

# Check if instance ID is provided
if [[ -z "${instance_ids}" ]]; then
    echo "Error: Missing required instance IDs parameter."
    usage
    return 1
fi

# shellcheck disable=SC2086
response=$(aws ec2 terminate-instances \
    "--instance-ids" $instance_ids \
    --query 'TerminatingInstances[*].[InstanceId,CurrentState.Name]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports terminate-instances operation failed.$response"
    return 1
}

return 0
}

#####
# function ec2_delete_security_group
#
# This function deletes an Amazon Elastic Compute Cloud (Amazon EC2) security
# group.
#
# Parameters:
#     -i security_group_id - The ID of the security group to delete.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_delete_security_group() {

```

```
local security_group_id response
local option OPTARG # Required to use getopt command in a function.

# bashsupport disable=BP5008
function usage() {
    echo "function ec2_delete_security_group"
    echo "Deletes an Amazon Elastic Compute Cloud (Amazon EC2) security group."
    echo "  -i security_group_id - The ID of the security group to delete."
    echo ""
}

# Retrieve the calling parameters.
while getopt "i:h" option; do
    case "${option}" in
        i) security_group_id="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -i parameter."
    usage
    return 1
fi

response=$(aws ec2 delete-security-group --group-id "$security_group_id" --
output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-security-group operation failed.$response"
    return 1
}

return 0
}
```

```
#####
# function ec2_delete_keypair
#
# This function deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair.
#
# Parameters:
#     -n key_pair_name - A key pair name.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_delete_keypair() {
    local key_pair_name response

    local option OPTARG # Required to use getopt command in a function.
    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_delete_keypair"
        echo "Deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair."
        echo "  -n key_pair_name - A key pair name."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:h" option; do
        case "${option}" in
            n) key_pair_name="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$key_pair_name" ]]; then
        errecho "ERROR: You must provide a key pair name with the -n parameter."
        usage
    fi
}
```

```

    return 1
fi

response=$(aws ec2 delete-key-pair \
  --key-name "$key_pair_name") || {
  aws_cli_error_log ${?}
  errecho "ERROR: AWS reports delete-key-pair operation failed.$response"
  return 1
}

return 0
}

```

Die in diesem Szenario verwendeten Dienstprogrammfunktionen.

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
  local err_code=$1
  errecho "Error code : $err_code"
  if [ "$err_code" == 1 ]; then
    errecho " One or more S3 transfers failed."
  elif [ "$err_code" == 2 ]; then

```

```
errecho " Command line failed to parse."
elif [ "$err_code" == 130 ]; then
    errecho " Process received SIGINT."
elif [ "$err_code" == 252 ]; then
    errecho " Command syntax invalid."
elif [ "$err_code" == 253 ]; then
    errecho " The system environment or configuration was invalid."
elif [ "$err_code" == 254 ]; then
    errecho " The service returned an error."
elif [ "$err_code" == 255 ]; then
    errecho " 255 is a catch-all error."
fi

return 0
}
```

- API-Einheiten finden Sie in den folgenden Themen in der AWS CLI Befehlsreferenz.
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)
 - [DeleteSecurityGroup](#)
 - [DescribeImages](#)
 - [DescribeInstanceTypes](#)
 - [DescribeInstances](#)
 - [DescribeKeyPairs](#)
 - [DescribeSecurityGroups](#)
 - [DisassociateAddress](#)
 - [ReleaseAddress](#)
 - [RunInstances](#)
 - [StartInstances](#)
 - [StopInstances](#)
 - [TerminateInstances](#)

- [UnmonitorInstances](#)

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Führen Sie ein Szenario an einer Eingabeaufforderung aus.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DescribeKeyPairsResponse;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressResponse;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressResponse;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse;
import software.amazon.awssdk.services.ssm.model.Parameter;

import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * 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 example performs the following tasks:
```

```
*
* 1. Creates an RSA key pair and saves the private key data as a .pem file.
* 2. Lists key pairs.
* 3. Creates a security group for the default VPC.
* 4. Displays security group information.
* 5. Gets a list of Amazon Linux 2 AMIs and selects one.
* 6. Gets additional information about the image.
* 7. Gets a list of instance types that are compatible with the selected AMI's
* architecture.
* 8. Creates an instance with the key pair, security group, AMI, and an
* instance type.
* 9. Displays information about the instance.
* 10. Stops the instance and waits for it to stop.
* 11. Starts the instance and waits for it to start.
* 12. Allocates an Elastic IP address and associates it with the instance.
* 13. Displays SSH connection info for the instance.
* 14. Disassociates and deletes the Elastic IP address.
* 15. Terminates the instance and waits for it to terminate.
* 16. Deletes the security group.
* 17. Deletes the key pair.
*/
public class EC2Scenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
    "-");
    private static final Logger logger =
    LoggerFactory.getLogger(EC2Scenario.class);
    public static void main(String[] args) throws InterruptedException,
    UnknownHostException {

        logger.info("""
        Usage:
            <keyName> <fileName> <groupName> <groupDesc>

        Where:
            keyName - A key pair name (for example, TestKeyPair).\s
            fileName - A file name where the key information is written to.\s
            groupName - The name of the security group.\s
            groupDesc - The description of the security group.\s
        """);

        Scanner scanner = new Scanner(System.in);
        EC2Actions ec2Actions = new EC2Actions();
```

```
String keyName = "TestKeyPair7" ;
String fileName = "ec2Key.pem";
String groupName = "TestSecGroup7" ;
String groupDesc = "Test Group" ;
String vpcId = ec2Actions.describeFirstEC2VpcAsync().join().vpcId();
InetAddress localAddress = InetAddress.getLocalHost();
String myIpAddress = localAddress.getHostAddress();

logger.info("""
    Amazon Elastic Compute Cloud (EC2) is a web service that provides
secure, resizable compute
    capacity in the cloud. It allows developers and organizations to
easily launch and manage
    virtual server instances, known as EC2 instances, to run their
applications.

    EC2 provides a wide range of instance types, each with different
compute, memory,
    and storage capabilities, to meet the diverse needs of various
workloads. Developers
    can choose the appropriate instance type based on their application's
requirements,
    such as high-performance computing, memory-intensive tasks, or GPU-
accelerated workloads.

    The `Ec2AsyncClient` interface in the AWS SDK for Java 2.x provides a
set of methods to
    programmatically interact with the Amazon EC2 service. This allows
developers to
    automate the provisioning, management, and monitoring of EC2
instances as part of their
    application deployment pipelines. With EC2, teams can focus on
building and deploying
    their applications without having to worry about the underlying
infrastructure
    required to host and manage physical servers.

    This scenario walks you through how to perform key operations for
this service.
    Let's get started...
    """);

waitForInputToContinue(scanner);
logger.info(DASHES);
```



```
        logger.info(DASHES);
        logger.info("1. Create an RSA key pair and save the private key material
as a .pem file.");
        logger.info("""
            An RSA key pair for Amazon EC2 is a security mechanism used to
authenticate and secure
            access to your EC2 instances. It consists of a public key and a
private key,
            which are generated as a pair.
            """);
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<CreateKeyPairResponse> future =
ec2Actions.createKeyPairAsync(keyName, fileName);
            CreateKeyPairResponse response = future.join();
            logger.info("Key Pair successfully created. Key Fingerprint: " +
response.keyFingerprint());

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                if (ec2Ex.getMessage().contains("already exists")) {
                    // Key pair already exists.
                    logger.info("The key pair '" + keyName + "' already exists.
Moving on...");
                } else {
                    logger.info("EC2 error occurred: Error message: {}, Error
code {}", ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                    return;
                }
            } else {
                logger.info("An unexpected error occurred: " +
(rt.getMessage()));
                return;
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("2. List key pairs.");
        waitForInputToContinue(scanner);
        try {
```

```
        CompletableFuture<DescribeKeyPairsResponse> future =
ec2Actions.describeKeysAsync();
        DescribeKeyPairsResponse keyPairsResponse = future.join();
        keyPairsResponse.keyPairs().forEach(keyPair -> logger.info(
            "Found key pair with name {} and fingerprint {}",
            keyPair.keyName(),
            keyPair.keyFingerprint()));

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof Ec2Exception ec2Ex) {
            logger.info("EC2 error occurred: Error message: {}, Error code
{}", ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
            return;
        } else {
            logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rt.getMessage()));
            return;
        }
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("3. Create a security group.");
    logger.info("""
        An AWS EC2 Security Group is a virtual firewall that controls the
        inbound and outbound traffic to an EC2 instance. It acts as a first
line
        of defense for your EC2 instances, allowing you to specify the rules
that
        govern the network traffic entering and leaving your instances.
        """);
    waitForInputToContinue(scanner);
    String groupId = "";
    try {
        CompletableFuture<String> future =
ec2Actions.createSecurityGroupAsync(groupName, groupDesc, vpcId, myIpAddress);
        future.join();
        logger.info("Created security group" );

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof Ec2Exception ec2Ex) {
```

```
        if (ec2Ex.awsErrorDetails().errorMessage().contains("already
exists")) {
            logger.info("The Security Group already exists. Moving
on...");
        } else {
            logger.error("An unexpected error occurred: {}",
ec2Ex.awsErrorDetails().errorMessage());
            return;
        }
    } else {
        logger.error("An unexpected error occurred: {}",
cause.getMessage());
        return;
    }
}
waitForInputToContinue(scanner);
logger.info(DASHES);

logger.info(DASHES);
logger.info("4. Display security group information for the new security
group.");
waitForInputToContinue(scanner);
try {
    CompletableFuture<String> future =
ec2Actions.describeSecurityGroupArnByNameAsync(groupName);
    groupId = future.join();
    logger.info("The security group Id is "+groupId);

} catch (RuntimeException rt) {
    Throwable cause = rt.getCause();
    if (cause instanceof Ec2Exception ec2Ex) {
        String errorCode = ec2Ex.awsErrorDetails().errorCode();
        if ("InvalidGroup.NotFound".equals(errorCode)) {
            logger.info("Security group '{}' does not exist. Error Code:
{}", groupName, errorCode);
        } else {
            logger.info("EC2 error occurred: Message {}, Error Code: {}",
ec2Ex.getMessage(), errorCode);
        }
    } else {
        logger.info("An unexpected error occurred: {}",
cause.getMessage());
    }
}
```

```
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("5. Get a list of Amazon Linux 2 AMIs and select one with
amzn2 in the name.");
        logger.info("""
            An Amazon EC2 AMI (Amazon Machine Image) is a pre-configured virtual
machine image that
            serves as a template for launching EC2 instances. It contains all the
necessary software and
            configurations required to run an application or operating system on
an EC2 instance.
            """);
        waitForInputToContinue(scanner);
        String instanceAMI="";
        try {
            CompletableFuture<GetParametersByPathResponse> future =
ec2Actions.getParaValuesAsync();
            GetParametersByPathResponse pathResponse = future.join();
            List<Parameter> parameterList = pathResponse.parameters();
            for (Parameter para : parameterList) {
                if (filterName(para.name())) {
                    instanceAMI = para.value();
                    break;
                }
            }
        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof Ec2Exception ec2Ex) {
                logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                return;
            } else {
                logger.info("An unexpected error occurred: {}",
cause.getMessage());
                return;
            }
        }
        logger.info("The AMI value with amzn2 is: {}", instanceAMI);
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
```

```
logger.info("6. Get the (Amazon Machine Image) AMI value from the amzn2
image.");
logger.info("""
    An AMI value represents a specific version of a virtual machine (VM)
or server image.
    It uniquely identifies a particular version of an EC2 instance,
including its operating system,
    pre-installed software, and any custom configurations. This allows you
to consistently deploy the same
    VM image across your infrastructure.

    """);
waitForInputToContinue(scanner);
String amiValue;
try {
    CompletableFuture<String> future =
ec2Actions.describeImageAsync(instanceAMI);
    amiValue = future.join();

} catch (CompletionException ce) {
    Throwable cause = ce.getCause();
    if (cause instanceof Ec2Exception) {
        Ec2Exception ec2Ex = (Ec2Exception) cause;
        logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
        return;
    } else {
        logger.info("An unexpected error occurred: {}",
cause.getMessage());
        return;
    }
}
waitForInputToContinue(scanner);
logger.info(DASHES);

logger.info(DASHES);
logger.info("7. Retrieves an instance type available in the current AWS
region.");
waitForInputToContinue(scanner);
String instanceType;
try {
    CompletableFuture<String> future =
ec2Actions.getInstanceTypesAsync();
    instanceType = future.join();
```

```
        if (!instanceType.isEmpty()) {
            logger.info("Found instance type: " + instanceType);
        } else {
            logger.info("Desired instance type not found.");
        }
    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof Ec2Exception ec2Ex) {
            logger.info("EC2 error occurred: Message {}, Error Code:{}",
                ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
            return;
        } else {
            logger.info("An unexpected error occurred: {}",
                cause.getMessage());
            return;
        }
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("8. Create an Amazon EC2 instance using the key pair, the
instance type, the security group, and the EC2 AMI value.");
    logger.info("Once the EC2 instance is created, it is placed into a
running state.");
    waitForInputToContinue(scanner);
    String newInstanceId;
    try {
        CompletableFuture<String> future =
ec2Actions.runInstanceAsync(instanceType, keyName, groupName, amiValue);
        newInstanceId = future.join();
    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof Ec2Exception) {
            Ec2Exception ec2Ex = (Ec2Exception) cause;
            switch (ec2Ex.awsErrorDetails().errorCode()) {
                case "InvalidParameterValue":
                    logger.info("EC2 error occurred: Message {}, Error Code:
{}", ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
                    break;
                case "InsufficientInstanceCapacity":
                    // Handle insufficient instance capacity.
                    logger.info("Insufficient instance capacity: {}, {}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
```

```
        break;
        case "InvalidGroup.NotFound":
            // Handle security group not found.
            logger.info("Security group not found: {},{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
            break;
        default:
            logger.info("EC2 error occurred: {} (Code: {})",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
            break;
    }
    return;
} else {
    logger.info("An unexpected error occurred: {}", (cause != null ?
cause.getMessage() : rt.getMessage()));
    return;
}
}
logger.info("The instance Id is " + newInstanceId);
waitForInputToContinue(scanner);
logger.info(DASHES);

logger.info(DASHES);
logger.info("9. Display information about the running instance. ");

waitForInputToContinue(scanner);
String publicIp;
try {
    CompletableFuture<String> future =
ec2Actions.describeEC2InstancesAsync(newInstanceId);
    publicIp = future.join();
    logger.info("EC2 instance public IP {}", publicIp);
} catch (RuntimeException rt) {
    Throwable cause = rt.getCause();
    if (cause instanceof Ec2Exception ec2Ex) {
        logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
        return;
    } else {
        logger.info("An unexpected error occurred: {}",
cause.getMessage());
        return;
    }
}
```

```
    }
    logger.info("You can SSH to the instance using this command:");
    logger.info("ssh -i " + fileName + " ec2-user@" + publicIp);
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("10. Stop the instance using a waiter (this may take a few
mins).");
    // Remove the 2nd one
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<Void> future =
ec2Actions.stopInstanceAsync(newInstanceId);
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof Ec2Exception ec2Ex) {
            logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
            return;
        } else {
            logger.info("An unexpected error occurred: {}",
cause.getMessage());
            return;
        }
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("11. Start the instance using a waiter (this may take a few
mins).");
    try {
        CompletableFuture<Void> future =
ec2Actions.startInstanceAsync(newInstanceId);
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof Ec2Exception ec2Ex) {
            // Handle EC2 exceptions.
```



```
        logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
        return;
    } else {
        logger.info("An unexpected error occurred: {}",
cause.getMessage());
        return;
    }
}
waitForInputToContinue(scanner);
logger.info(DASHES);

logger.info(DASHES);
logger.info("12. Allocate an Elastic IP address and associate it with the
instance.");
logger.info("""
    An Elastic IP address is a static public IP address that you can
associate with your EC2 instance.
    This allows you to have a fixed, predictable IP address that remains
the same even if your instance
    is stopped, terminated, or replaced.
    This is particularly useful for applications or services that need to
be accessed consistently from a
    known IP address.

    An EC2 Allocation ID (also known as a Reserved Instance Allocation
ID) is a unique identifier associated with a Reserved Instance (RI) that you
have purchased in AWS.

    When you purchase a Reserved Instance, AWS assigns a unique
Allocation ID to it.
    This Allocation ID is used to track and identify the specific RI you
have purchased,
    and it is important for managing and monitoring your Reserved
Instances.

    """);

waitForInputToContinue(scanner);
String allocationId;
try {
    CompletableFuture<String> future = ec2Actions.allocateAddressAsync();
    allocationId = future.join();
```

```
        logger.info("Successfully allocated address with ID: "
+allocationId);
    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof Ec2Exception ec2Ex) {
            logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
            return;
        } else {
            logger.info("An unexpected error occurred: {}",
cause.getMessage());
            return;
        }
    }
    logger.info("The allocation Id value is " + allocationId);
    waitForInputToContinue(scanner);
    String associationId;
    try {
        CompletableFuture<String> future =
ec2Actions.associateAddressAsync(newInstanceId, allocationId);
        associationId = future.join();
        logger.info("Successfully associated address with ID: "
+associationId);
    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof Ec2Exception ec2Ex) {
            logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
            return;
        } else {
            logger.info("An unexpected error occurred: {}",
cause.getMessage());
            return;
        }
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("13. Describe the instance again. Note that the public IP
address has changed");
    waitForInputToContinue(scanner);
    try {
```

```
        CompletableFuture<String> future =
ec2Actions.describeEC2InstancesAsync(newInstanceId);
        publicIp = future.join();
        logger.info("EC2 instance public IP: " + publicIp);
        logger.info("You can SSH to the instance using this command:");
        logger.info("ssh -i " + fileName + " ec2-user@" + publicIp);
    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof Ec2Exception ec2Ex) {
            logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
            return;
        } else {
            logger.info("An unexpected error occurred: {}",
cause.getMessage());
            return;
        }
    }
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("14. Disassociate and release the Elastic IP address.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<DisassociateAddressResponse> future =
ec2Actions.disassociateAddressAsync(associationId);
        future.join();
        logger.info("Address successfully disassociated.");
    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof Ec2Exception ec2Ex) {
            // Handle EC2 exceptions.
            logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
            return;
        } else {
            logger.info("An unexpected error occurred: {}",
cause.getMessage());
            return;
        }
    }
    }
    waitForInputToContinue(scanner);
    try {
```

```
        CompletableFuture<ReleaseAddressResponse> future =
ec2Actions.releaseEC2AddressAsync(allocationId);
        future.join(); // Wait for the operation to complete
        logger.info("Elastic IP address successfully released.");
    } catch (RuntimeException rte) {
        logger.info("An unexpected error occurred: {}", rte.getMessage());
        return;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("15. Terminate the instance and use a waiter (this may take a
few mins).");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<Object> future =
ec2Actions.terminateEC2Async(newInstanceId);
        future.join();
        logger.info("EC2 instance successfully terminated.");
    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof Ec2Exception ec2Ex) {
            // Handle EC2 exceptions.
            logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
            return;
        } else {
            logger.info("An unexpected error occurred: {}",
cause.getMessage());
            return;
        }
    }
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("16. Delete the security group.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<Void> future =
ec2Actions.deleteEC2SecGroupAsync(groupId);
        future.join();
        logger.info("Security group successfully deleted.");
    } catch (RuntimeException rt) {
```

```
        Throwable cause = rt.getCause();
        if (cause instanceof Ec2Exception ec2Ex) {
            logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
            return;
        } else {
            logger.info("An unexpected error occurred: {}",
cause.getMessage());
            return;
        }
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("17. Delete the key.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<DeleteKeyPairResponse> future =
ec2Actions.deleteKeysAsync(keyName);
        future.join();
        logger.info("Successfully deleted key pair named " + keyName);
    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof Ec2Exception ec2Ex) {
            logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
            return;
        } else {
            logger.info("An unexpected error occurred: {}",
cause.getMessage());
            return;
        }
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("You successfully completed the Amazon EC2 scenario.");
    logger.info(DASHES);
}
public static boolean filterName(String name) {
    String[] parts = name.split("/");
    String myValue = parts[4];
```

```
        return myValue.contains("amzn2");
    }

    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.");
            }
        }
    }
}
```

Definiere eine Klasse, die EC2 Aktionen umschließt.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
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.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.AllocateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AllocateAddressResponse;
import software.amazon.awssdk.services.ec2.model.AssociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AssociateAddressResponse;
import
    software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupRequest;
```

```
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstanceTypesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstanceTypesResponse;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeKeyPairsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsRequest;
import software.amazon.awssdk.services.ec2.model.DescribeSecurityGroupsResponse;
import software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.DisassociateAddressResponse;
import software.amazon.awssdk.services.ec2.model.DomainType;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.model.Filter;
import software.amazon.awssdk.services.ec2.model.InstanceTypeInfo;
import software.amazon.awssdk.services.ec2.model.IpPermission;
import software.amazon.awssdk.services.ec2.model.IpRange;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressRequest;
import software.amazon.awssdk.services.ec2.model.ReleaseAddressResponse;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.StopInstancesRequest;
import software.amazon.awssdk.services.ec2.model.StartInstancesRequest;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesRequest;
import software.amazon.awssdk.services.ec2.model.Vpc;
import software.amazon.awssdk.services.ec2.paginators.DescribeImagesPublisher;
import software.amazon.awssdk.services.ec2.paginators.DescribeInstancesPublisher;
import
    software.amazon.awssdk.services.ec2.paginators.DescribeSecurityGroupsPublisher;
import software.amazon.awssdk.services.ec2.paginators.DescribeVpcsPublisher;
import software.amazon.awssdk.services.ec2.waiters.Ec2AsyncWaiter;
import software.amazon.awssdk.services.ssm.SsmAsyncClient;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathRequest;
import software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse;
import software.amazon.awssdk.services.ec2.model.TerminateInstancesResponse;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;
import java.util.concurrent.atomic.AtomicReference;
```

```
public class EC2Actions {
    private static final Logger logger =
    LoggerFactory.getLogger(EC2Actions.class);
    private static Ec2AsyncClient ec2AsyncClient;

    /**
     * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
     *
     * @return the configured ECR asynchronous client.
     */
    private static Ec2AsyncClient getAsyncClient() {
        if (ec2AsyncClient == 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.
                .build();

            ec2AsyncClient = Ec2AsyncClient.builder()
                .region(Region.US_EAST_1)
                .httpClient(httpClient)
                .overrideConfiguration(overrideConfig)
                .build();
        }
    }
}
```



```
        return ec2AsyncClient;
    }

    /**
     * Deletes a key pair asynchronously.
     *
     * @param keyPair the name of the key pair to delete
     * @return a {@link CompletableFuture} that represents the result of the
     asynchronous operation.
     *
     * The {@link CompletableFuture} will complete with a {@link
     DeleteKeyPairResponse} object
     *
     * that provides the result of the key pair deletion operation.
     */
    public CompletableFuture<DeleteKeyPairResponse> deleteKeysAsync(String
    keyPair) {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
            .keyName(keyPair)
            .build();

        // Initiate the asynchronous request to delete the key pair.
        CompletableFuture<DeleteKeyPairResponse> response =
    getAsyncClient().deleteKeyPair(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete key pair: " +
    keyPair, ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting key
    pair: " + keyPair);
            }
        });
    }

    /**
     * Deletes an EC2 security group asynchronously.
     *
     * @param groupId the ID of the security group to delete
     * @return a CompletableFuture that completes when the security group is
     deleted
     */
    public CompletableFuture<Void> deleteEC2SecGroupAsync(String groupId) {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();
    }
```

```
        CompletableFuture<DeleteSecurityGroupResponse> response =
getAsyncClient().deleteSecurityGroup(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to delete security group with
Id " + groupId, ex);
            } else if (resp == null) {
                throw new RuntimeException("No response received for deleting
security group with Id " + groupId);
            }
        }).thenApply(resp -> null);
    }

/**
 * Terminates an EC2 instance asynchronously and waits for it to reach the
terminated state.
 *
 * @param instanceId the ID of the EC2 instance to terminate
 * @return a {@link CompletableFuture} that completes when the instance has
been terminated
 * @throws RuntimeException if there is no response from the AWS SDK or if
there is a failure during the termination process
 */
    public CompletableFuture<Object> terminateEC2Async(String instanceId) {
        TerminateInstancesRequest terminateRequest =
TerminateInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        CompletableFuture<TerminateInstancesResponse> responseFuture =
getAsyncClient().terminateInstances(terminateRequest);
        return responseFuture.thenCompose(terminateResponse -> {
            if (terminateResponse == null) {
                throw new RuntimeException("No response received for terminating
instance " + instanceId);
            }
            System.out.println("Going to terminate an EC2 instance and use a
waiter to wait for it to be in terminated state");
            return getAsyncClient().waiter()
                .waitUntilInstanceTerminated(r -> r.instanceIds(instanceId))
                .thenApply(waiterResponse -> null);
        }).exceptionally(throwable -> {
            // Handle any exceptions that occurred during the async call

```

```
        throw new RuntimeException("Failed to terminate EC2 instance: " +
throwable.getMessage(), throwable);
    });
}

/**
 * Releases an Elastic IP address asynchronously.
 *
 * @param allocId the allocation ID of the Elastic IP address to be released
 * @return a {@link CompletableFuture} representing the asynchronous
operation of releasing the Elastic IP address
 */
public CompletableFuture<ReleaseAddressResponse>
releaseEC2AddressAsync(String allocId) {
    ReleaseAddressRequest request = ReleaseAddressRequest.builder()
        .allocationId(allocId)
        .build();

    CompletableFuture<ReleaseAddressResponse> response =
getAsyncClient().releaseAddress(request);
    response.whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to release Elastic IP
address", ex);
        }
    });

    return response;
}

/**
 * Disassociates an Elastic IP address from an instance asynchronously.
 *
 * @param associationId The ID of the association you want to disassociate.
 * @return a {@link CompletableFuture} representing the asynchronous
operation of disassociating the address. The
 *       {@link CompletableFuture} will complete with a {@link
DisassociateAddressResponse} when the operation is
 *       finished.
 * @throws RuntimeException if the disassociation of the address fails.
 */
public CompletableFuture<DisassociateAddressResponse>
disassociateAddressAsync(String associationId) {
    Ec2AsyncClient ec2 = getAsyncClient();
```

```
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
        .associationId(associationId)
        .build();

        // Disassociate the address asynchronously.
        CompletableFuture<DisassociateAddressResponse> response =
ec2.disassociateAddress(addressRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to disassociate address", ex);
            }
        });

        return response;
    }

    /**
     * Associates an Elastic IP address with an EC2 instance asynchronously.
     *
     * @param instanceId    the ID of the EC2 instance to associate the Elastic
Elastic IP address with
     * @param allocationId  the allocation ID of the Elastic IP address to
associate
     * @return a {@link CompletableFuture} that completes with the association ID
when the operation is successful,
     *         or throws a {@link RuntimeException} if the operation fails
     */
    public CompletableFuture<String> associateAddressAsync(String instanceId,
String allocationId) {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
        .instanceId(instanceId)
        .allocationId(allocationId)
        .build();

        CompletableFuture<AssociateAddressResponse> responseFuture =
getAsyncClient().associateAddress(associateRequest);
        return responseFuture.thenApply(response -> {
            if (response.associationId() != null) {
                return response.associationId();
            } else {
                throw new RuntimeException("Association ID is null after
associating address.");
            }
        });
    }
}
```

```
    }
    }).whenComplete((result, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to associate address", ex);
        }
    });
}

/**
 * Allocates an Elastic IP address asynchronously in the VPC domain.
 *
 * @return a {@link CompletableFuture} containing the allocation ID of the
allocated Elastic IP address
 */
public CompletableFuture<String> allocateAddressAsync() {
    AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
        .domain(DomainType.VPC)
        .build();

    CompletableFuture<AllocateAddressResponse> responseFuture =
getAsyncClient().allocateAddress(allocateRequest);
    return
responseFuture.thenApply(AllocateAddressResponse::allocationId).whenComplete((result,
ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to allocate address", ex);
        }
    });
}

/**
 * Asynchronously describes the state of an EC2 instance.
 * The paginator helps you iterate over multiple pages of results.
 *
 * @param newInstanceId the ID of the EC2 instance to describe
 * @return a {@link CompletableFuture} that, when completed, contains a
string describing the state of the EC2 instance
 */
public CompletableFuture<String> describeEC2InstancesAsync(String
newInstanceId) {
    DescribeInstancesRequest request = DescribeInstancesRequest.builder()
        .instanceIds(newInstanceId)
        .build();
```

```
DescribeInstancesPublisher paginator =
getAsyncClient().describeInstancesPaginator(request);
AtomicReference<String> publicIpAddressRef = new AtomicReference<>();
return paginator.subscribe(response -> {
    response.reservations().stream()
        .flatMap(reservation -> reservation.instances().stream())
        .filter(instance -> instance.instanceId().equals(newInstanceId))
        .findFirst()
        .ifPresent(instance ->
publicIpAddressRef.set(instance.publicIpAddress()));
    }).thenApply(v -> {
        String publicIpAddress = publicIpAddressRef.get();
        if (publicIpAddress == null) {
            throw new RuntimeException("Instance with ID " + newInstanceId +
" not found.");
        }
        return publicIpAddress;
    }).exceptionally(ex -> {
        logger.info("Failed to describe instances: " + ex.getMessage());
        throw new RuntimeException("Failed to describe instances", ex);
    });
}

/**
 * Runs an EC2 instance asynchronously.
 *
 * @param instanceType The instance type to use for the EC2 instance.
 * @param keyName The name of the key pair to associate with the EC2
instance.
 * @param groupName The name of the security group to associate with the EC2
instance.
 * @param amiId The ID of the Amazon Machine Image (AMI) to use for the EC2
instance.
 * @return A {@link CompletableFuture} that completes with the ID of the
started EC2 instance.
 * @throws RuntimeException If there is an error running the EC2 instance.
 */
public CompletableFuture<String> runInstanceAsync(String instanceType, String
keyName, String groupName, String amiId) {
    RunInstancesRequest runRequest = RunInstancesRequest.builder()
        .instanceType(instanceType)
        .keyName(keyName)
        .securityGroups(groupName)
        .maxCount(1)
```

```
        .minCount(1)
        .imageId(amiId)
        .build();

    CompletableFuture<RunInstancesResponse> responseFuture =
getAsyncClient().runInstances(runRequest);
    return responseFuture.thenCompose(response -> {
        String instanceIdVal = response.instances().get(0).instanceId();
        System.out.println("Going to start an EC2 instance and use a waiter
to wait for it to be in running state");
        return getAsyncClient().waiter()
            .waitUntilInstanceExists(r -> r.instanceIds(instanceIdVal))
            .thenCompose(waitResponse -> getAsyncClient().waiter()
                .waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal))
                .thenApply(runningResponse -> instanceIdVal));
    }).exceptionally(throwable -> {
        // Handle any exceptions that occurred during the async call
        throw new RuntimeException("Failed to run EC2 instance: " +
throwable.getMessage(), throwable);
    });
}

/**
 * Asynchronously retrieves the instance types available in the current AWS
region.
 * <p>
 * This method uses the AWS SDK's asynchronous API to fetch the available
instance types
 * and then processes the response. It logs the memory information, network
information,
 * and instance type for each instance type returned. Additionally, it
returns a
 * {@link CompletableFuture} that resolves to the instance type string for
the "t2.2xlarge"
 * instance type, if it is found in the response. If the "t2.2xlarge"
instance type is not
 * found, an empty string is returned.
 * </p>
 *
 * @return a {@link CompletableFuture} that resolves to the instance type
string for the
 * "t2.2xlarge" instance type, or an empty string if the instance type is not
found
 */
```

```
public CompletableFuture<String> getInstanceTypesAsync() {
    DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
        .maxResults(10)
        .build();

    CompletableFuture<DescribeInstanceTypesResponse> response =
getAsyncClient().describeInstanceTypes(typesRequest);
    response.whenComplete((resp, ex) -> {
        if (resp != null) {
            List<InstanceTypeInfo> instanceTypes = resp.instanceTypes();
            for (InstanceTypeInfo type : instanceTypes) {
                logger.info("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
                logger.info("Network information is " +
type.networkInfo().toString());
                logger.info("Instance type is " +
type.instanceType().toString());
            }
        } else {
            throw (RuntimeException) ex;
        }
    });

    return response.thenApply(resp -> {
        for (InstanceTypeInfo type : resp.instanceTypes()) {
            String instanceType = type.instanceType().toString();
            if (instanceType.equals("t2.2xlarge")) {
                return instanceType;
            }
        }
        return "";
    });
}

/**
 * Asynchronously describes an AWS EC2 image with the specified image ID.
 *
 * @param imageId the ID of the image to be described
 * @return a {@link CompletableFuture} that, when completed, contains the ID
of the described image
 * @throws RuntimeException if no images are found with the provided image
ID, or if an error occurs during the AWS API call
 */
}
```



```

public CompletableFuture<String> describeImageAsync(String imageId) {
    DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
        .imageIds(imageId)
        .build();

    AtomicReference<String> imageIdRef = new AtomicReference<>();
    DescribeImagesPublisher paginator =
getAsyncClient().describeImagesPaginator(imagesRequest);
    return paginator.subscribe(response -> {
        response.images().stream()
            .filter(image -> image.imageId().equals(imageId))
            .findFirst()
            .ifPresent(image -> {
                logger.info("The description of the image is " +
image.description());
                logger.info("The name of the image is " + image.name());
                imageIdRef.set(image.imageId());
            });
    }).thenApply(v -> {
        String id = imageIdRef.get();
        if (id == null) {
            throw new RuntimeException("No images found with the provided
image ID.");
        }
        return id;
    }).exceptionally(ex -> {
        logger.info("Failed to describe image: " + ex.getMessage());
        throw new RuntimeException("Failed to describe image", ex);
    });
}

/**
 * Retrieves the parameter values asynchronously using the AWS Systems
Manager (SSM) API.
 *
 * @return a {@link CompletableFuture} that holds the response from the SSM
API call to get parameters by path
 */
public CompletableFuture<GetParametersByPathResponse> getParaValuesAsync() {
    SsmAsyncClient ssmClient = SsmAsyncClient.builder()
        .region(Region.US_EAST_1)
        .build();
}

```

```
    GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
    .path("/aws/service/ami-amazon-linux-latest")
    .build();

    // Create a CompletableFuture to hold the final result.
    CompletableFuture<GetParametersByPathResponse> responseFuture = new
CompletableFuture<>();
    ssmClient.getParametersByPath(parameterRequest)
    .whenComplete((response, exception) -> {
        if (exception != null) {
            responseFuture.completeExceptionally(new
RuntimeException("Failed to get parameters by path", exception));
        } else {
            responseFuture.complete(response);
        }
    });

    return responseFuture;
}

/**
 * Asynchronously describes the security groups for the specified group ID.
 *
 * @param groupName the name of the security group to describe
 * @return a {@link CompletableFuture} that represents the asynchronous
operation
 *
 * of describing the security groups. The future will complete with a
 *
 * {@link DescribeSecurityGroupsResponse} object that contains the
 *
 * security group information.
 */
public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
    DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
    .groupNames(groupName)
    .build();

    DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
    AtomicReference<String> groupIdRef = new AtomicReference<>();
    return paginator.subscribe(response -> {
        response.securityGroups().stream()
```

```
        .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
        .findFirst()
        .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
    }).thenApply(v -> {
        String groupId = groupIdRef.get();
        if (groupId == null) {
            throw new RuntimeException("No security group found with the
name: " + groupName);
        }
        return groupId;
    }).exceptionally(ex -> {
        logger.info("Failed to describe security group: " + ex.getMessage());
        throw new RuntimeException("Failed to describe security group", ex);
    });
}

/**
 * Creates a new security group asynchronously with the specified group name,
description, and VPC ID. It also
 * authorizes inbound traffic on ports 80 and 22 from the specified IP
address.
 *
 * @param groupName    the name of the security group to create
 * @param groupDesc    the description of the security group
 * @param vpcId        the ID of the VPC in which to create the security
group
 * @param myIpAddress  the IP address from which to allow inbound traffic
(e.g., "192.168.1.1/0" to allow traffic from
 *                    any IP address in the 192.168.1.0/24 subnet)
 * @return a CompletableFuture that, when completed, returns the ID of the
created security group
 * @throws RuntimeException if there was a failure creating the security
group or authorizing the inbound traffic
 */
public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
    CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
        .groupName(groupName)
        .description(groupDesc)
        .vpcId(vpcId)
        .build();
```

```
return getAsyncClient().createSecurityGroup(createRequest)
    .thenCompose(createResponse -> {
        String groupId = createResponse.groupId();
        IpRange ipRange = IpRange.builder()
            .cidrIp(myIpAddress + "/32")
            .build();

        IpPermission ipPerm = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(80)
            .fromPort(80)
            .ipRanges(ipRange)
            .build();

        IpPermission ipPerm2 = IpPermission.builder()
            .ipProtocol("tcp")
            .toPort(22)
            .fromPort(22)
            .ipRanges(ipRange)
            .build();

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(groupId)
            .ipPermissions(ipPerm, ipPerm2)
            .build();

        return
getAsyncClient().authorizeSecurityGroupIngress(authRequest)
            .thenApply(authResponse -> groupId);
    })
    .whenComplete((result, exception) -> {
        if (exception != null) {
            if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                throw (Ec2Exception) exception.getCause();
            } else {
                throw new RuntimeException("Failed to create security
group: " + exception.getMessage(), exception);
            }
        }
    });
}
```

```
/**
 * Asynchronously describes the key pairs associated with the current AWS
 account.
 *
 * @return a {@link CompletableFuture} containing the {@link
 DescribeKeyPairsResponse} object, which provides
 * information about the key pairs.
 */
public CompletableFuture<DescribeKeyPairsResponse> describeKeysAsync() {
    CompletableFuture<DescribeKeyPairsResponse> responseFuture =
getAsyncClient().describeKeyPairs();
    responseFuture.whenComplete((response, exception) -> {
        if (exception != null) {
            throw new RuntimeException("Failed to describe key pairs: " +
exception.getMessage(), exception);
        }
    });

    return responseFuture;
}

/**
 * Creates a new key pair asynchronously.
 *
 * @param keyName the name of the key pair to create
 * @param fileName the name of the file to write the key material to
 * @return a {@link CompletableFuture} that represents the asynchronous
 operation
 *         of creating the key pair and writing the key material to a file
 */
public CompletableFuture<CreateKeyPairResponse> createKeyPairAsync(String
keyName, String fileName) {
    CreateKeyPairRequest request = CreateKeyPairRequest.builder()
        .keyName(keyName)
        .build();

    CompletableFuture<CreateKeyPairResponse> responseFuture =
getAsyncClient().createKeyPair(request);
    responseFuture.whenComplete((response, exception) -> {
        if (response != null) {
            try {
                BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName));
```

```
        writer.write(response.keyMaterial());
        writer.close();
    } catch (IOException e) {
        throw new RuntimeException("Failed to write key material to
file: " + e.getMessage(), e);
    }
    } else {
        throw new RuntimeException("Failed to create key pair: " +
exception.getMessage(), exception);
    }
});

return responseFuture;
}

/**
 * Describes the first default VPC asynchronously and using a paginator.
 *
 * @return a {@link CompletableFuture} that, when completed, contains the
first default VPC found.
 */
public CompletableFuture<Vpc> describeFirstEC2VpcAsync() {
    Filter myFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    DescribeVpcsRequest request = DescribeVpcsRequest.builder()
        .filters(myFilter)
        .build();

    DescribeVpcsPublisher paginator =
getAsyncClient().describeVpcsPaginator(request);
    AtomicReference<Vpc> vpcRef = new AtomicReference<>();
    return paginator.subscribe(response -> {
        response.vpcs().stream()
            .findFirst()
            .ifPresent(vpcRef::set);
    }).thenApply(v -> {
        Vpc vpc = vpcRef.get();
        if (vpc == null) {
            throw new RuntimeException("Default VPC not found");
        }
        return vpc;
    });
}
```

```
    }).exceptionally(ex -> {
        logger.info("Failed to describe VPCs: " + ex.getMessage());
        throw new RuntimeException("Failed to describe VPCs", ex);
    });
}

/**
 * Stops the EC2 instance with the specified ID asynchronously and waits for
the instance to stop.
 *
 * @param instanceId the ID of the EC2 instance to stop
 * @return a {@link CompletableFuture} that completes when the instance has
been stopped, or exceptionally if an error occurs
 */
public CompletableFuture<Void> stopInstanceAsync(String instanceId) {
    StopInstancesRequest stopRequest = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
        .client(getAsyncClient())
        .build();

    CompletableFuture<Void> resultFuture = new CompletableFuture<>();
    logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
    getAsyncClient().stopInstances(stopRequest)
        .thenCompose(response -> {
            if (response.stoppingInstances().isEmpty()) {
                return CompletableFuture.failedFuture(new
RuntimeException("No instances were stopped. Please check the instance ID: " +
instanceId));
            }
            return ec2Waiter.waitUntilInstanceStopped(describeRequest);
        })
        .thenAccept(waiterResponse -> {
            logger.info("Successfully stopped instance " + instanceId);
            resultFuture.complete(null);
        })
}
```

```
        .exceptionally(throwable -> {
            logger.error("Failed to stop instance " + instanceId + ": " +
throwable.getMessage(), throwable);
            resultFuture.completeExceptionally(new RuntimeException("Failed
to stop instance: " + throwable.getMessage(), throwable));
            return null;
        });

    return resultFuture;
}

/**
 * Starts an Amazon EC2 instance asynchronously and waits until it is in the
"running" state.
 *
 * @param instanceId the ID of the instance to start
 * @return a {@link CompletableFuture} that completes when the instance has
been started and is in the "running" state, or exceptionally if an error occurs
 */
public CompletableFuture<Void> startInstanceAsync(String instanceId) {
    StartInstancesRequest startRequest = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
        .client(getAsyncClient())
        .build();

    DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    logger.info("Starting instance " + instanceId + " and waiting for it to
run.");
    CompletableFuture<Void> resultFuture = new CompletableFuture<>();
    return getAsyncClient().startInstances(startRequest)
        .thenCompose(response ->
            ec2Waiter.waitUntilInstanceRunning(describeRequest)
        )
        .thenAccept(waiterResponse -> {
            logger.info("Successfully started instance " + instanceId);
            resultFuture.complete(null);
        })
}
```



```
        .exceptionally(throwable -> {
            resultFuture.completeExceptionally(new RuntimeException("Failed
to start instance: " + throwable.getMessage(), throwable));
            return null;
        });
    }
}
```

- APIEinzelheiten finden Sie unter den folgenden Themen im AWS SDK for Java 2.x APIReferenzhandbuch.
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)
 - [DeleteSecurityGroup](#)
 - [DescribeImages](#)
 - [DescribeInstanceTypes](#)
 - [DescribeInstances](#)
 - [DescribeKeyPairs](#)
 - [DescribeSecurityGroups](#)
 - [DisassociateAddress](#)
 - [ReleaseAddress](#)
 - [RunInstances](#)
 - [StartInstances](#)
 - [StopInstances](#)
 - [TerminateInstances](#)
 - [UnmonitorInstances](#)

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Diese Datei enthält eine Liste gängiger Aktionen, die mit verwendet werden EC2. Die Schritte werden mit einem Szenario-Framework erstellt, das die Ausführung eines interaktiven Beispiels vereinfacht. Den vollständigen Kontext finden Sie im GitHub Repository.

```
import { tmpdir } from "node:os";
import { writeFile, mkdtemp, rm } from "node:fs/promises";
import { join } from "node:path";
import { get } from "node:http";

import {
  AllocateAddressCommand,
  AssociateAddressCommand,
  AuthorizeSecurityGroupIngressCommand,
  CreateKeyPairCommand,
  CreateSecurityGroupCommand,
  DeleteKeyPairCommand,
  DeleteSecurityGroupCommand,
  DisassociateAddressCommand,
  paginateDescribeImages,
  paginateDescribeInstances,
  paginateDescribeInstanceTypes,
  ReleaseAddressCommand,
  RunInstancesCommand,
  StartInstancesCommand,
  StopInstancesCommand,
  TerminateInstancesCommand,
  waitUntilInstanceStatusOk,
  waitUntilInstanceStopped,
  waitUntilInstanceTerminated,
} from "@aws-sdk/client-ec2";

import {
```

```
ScenarioAction,  
ScenarioInput,  
ScenarioOutput,  
} from "@aws-doc-sdk-examples/lib/scenario/index.js";  
  
import { paginateGetParametersByPath, SSMClient } from "@aws-sdk/client-ssm";  
  
/**  
 * @typedef {{  
 *   ec2Client: import('@aws-sdk/client-ec2').EC2Client,  
 *   errors: Error[],  
 *   keyPairId?: string,  
 *   tmpDirectory?: string,  
 *   securityGroupId?: string,  
 *   ipAddress?: string,  
 *   images?: import('@aws-sdk/client-ec2').Image[],  
 *   image?: import('@aws-sdk/client-ec2').Image,  
 *   instanceTypes?: import('@aws-sdk/client-ec2').InstanceTypeInfo[],  
 *   instanceId?: string,  
 *   instanceIpAddress?: string,  
 *   allocationId?: string,  
 *   allocatedIpAddress?: string,  
 *   associationId?: string,  
 * }} State  
 */  
  
/**  
 * A skip function provided to the `skipWhen` of a Step when you want  
 * to ignore that step if any errors have occurred.  
 * @param {State} state  
 */  
const skipWhenErrors = (state) => state.errors.length > 0;  
  
const MAX_WAITER_TIME_IN_SECONDS = 60 * 8;  
  
export const confirm = new ScenarioInput("confirmContinue", "Continue?", {  
  type: "confirm",  
  skipWhen: skipWhenErrors,  
});  
  
export const exitOnNoConfirm = new ScenarioAction(  
  "exitOnConfirmContinueFalse",  
  (/** @type { { earlyExit: boolean } & Record<string, any>} */ state) => {  
    if (!state[confirm.name]) {
```

```
    state.earlyExit = true;
  }
},
{
  skipWhen: skipWhenErrors,
},
);

export const greeting = new ScenarioOutput(
  "greeting",
  `

Welcome to the Amazon EC2 basic usage scenario.

Before you launch an instances, you'll need to provide a few things:
- A key pair - This is for SSH access to your EC2 instance. You only need to
provide the name.
- A security group - This is used for configuring access to your instance.
Again, only the name is needed.
- An IP address - Your public IP address will be fetched.
- An Amazon Machine Image (AMI)
- A compatible instance type`,
  { header: true, preformatted: true, skipWhen: skipWhenErrors },
);

export const provideKeyName = new ScenarioInput(
  "keyPairName",
  "Provide a name for a new key pair.",
  { type: "input", default: "ec2-example-key-pair", skipWhen: skipWhenErrors },
);

export const createKeyPair = new ScenarioAction(
  "createKeyPair",
  async (/** @type {State} */ state) => {
    try {
      // Create a key pair in Amazon EC2.
      const { KeyMaterial, KeyPairId } = await state.ec2Client.send(
        // A unique name for the key pair. Up to 255 ASCII characters.
        new CreateKeyPairCommand({ KeyName: state[provideKeyName.name] }),
      );

      state.keyPairId = KeyPairId;

      // Save the private key in a temporary location.
    }
  }
);
```

```
state.tmpDirectory = await mkdtemp(join(tmpdir(), "ec2-scenario-tmp"));
await writeFile(
  `${state.tmpDirectory}/${state[provideKeyPairName.name]}.pem`,
  KeyMaterial,
  {
    mode: 0o400,
  },
);
} catch (caught) {
  if (
    caught instanceof Error &&
    caught.name === "InvalidKeyPair.Duplicate"
  ) {
    caught.message = `${caught.message}. Try another key name.`;
  }

  state.errors.push(caught);
}
},
{ skipWhen: skipWhenErrors },
);

export const logKeyPair = new ScenarioOutput(
  "logKeyPair",
  (/** @type {State} */ state) =>
    `Created the key pair ${state[provideKeyPairName.name]}.\`,
  { skipWhen: skipWhenErrors },
);

export const confirmDeleteKeyPair = new ScenarioInput(
  "confirmDeleteKeyPair",
  "Do you want to delete the key pair?",
  {
    type: "confirm",
    // Don't do anything when a key pair was never created.
    skipWhen: (/** @type {State} */ state) => !state.keyPairId,
  },
);

export const maybeDeleteKeyPair = new ScenarioAction(
  "deleteKeyPair",
  async (/** @type {State} */ state) => {
    try {
      // Delete a key pair by name from EC2
```

```
    await state.ec2Client.send(
      new DeleteKeyPairCommand({ KeyName: state[provideKeyPairName.name] }),
    );
  } catch (caught) {
    if (
      caught instanceof Error &&
      // Occurs when a required parameter (e.g. KeyName) is undefined.
      caught.name === "MissingParameter"
    ) {
      caught.message = `${caught.message}. Did you provide the required value?`;
    }
    state.errors.push(caught);
  }
},
{
  // Don't do anything when there's no key pair to delete or the user chooses
  // to keep it.
  skipWhen: (/** @type {State} */ state) =>
    !state.keyPairId || !state[confirmDeleteKeyPair.name],
},
);

export const provideSecurityGroupName = new ScenarioInput(
  "securityGroupName",
  "Provide a name for a new security group.",
  { type: "input", default: "ec2-scenario-sg", skipWhen: skipWhenErrors },
);

export const createSecurityGroup = new ScenarioAction(
  "createSecurityGroup",
  async (/** @type {State} */ state) => {
    try {
      // Create a new security group that will be used to configure ingress/
      // egress for
      // an EC2 instance.
      const { GroupId } = await state.ec2Client.send(
        new CreateSecurityGroupCommand({
          GroupName: state[provideSecurityGroupName.name],
          Description: "A security group for the Amazon EC2 example.",
        }),
      );
      state.securityGroupId = GroupId;
    } catch (caught) {
```

```
    if (caught instanceof Error && caught.name === "InvalidGroup.Duplicate") {
      caught.message = `${caught.message}. Please provide a different name for
your security group.`;
    }

    state.errors.push(caught);
  }
},
{ skipWhen: skipWhenErrors },
);

export const logSecurityGroup = new ScenarioOutput(
  "logSecurityGroup",
  (/** @type {State} */ state) =>
    `Created the security group ${state.securityGroupId}.`,
  { skipWhen: skipWhenErrors },
);

export const confirmDeleteSecurityGroup = new ScenarioInput(
  "confirmDeleteSecurityGroup",
  "Do you want to delete the security group?",
  {
    type: "confirm",
    // Don't do anything when a security group was never created.
    skipWhen: (/** @type {State} */ state) => !state.securityGroupId,
  },
);

export const maybeDeleteSecurityGroup = new ScenarioAction(
  "deleteSecurityGroup",
  async (/** @type {State} */ state) => {
    try {
      // Delete the security group if the 'skipWhen' condition below is not met.
      await state.ec2Client.send(
        new DeleteSecurityGroupCommand({
          GroupId: state.securityGroupId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidGroupId.Malformed"
      ) {
        caught.message = `${caught.message}. Please provide a valid GroupId.`;
      }
    }
  }
);
```

```
    }
    state.errors.push(caught);
  }
},
{
  // Don't do anything when there's no security group to delete
  // or the user chooses to keep it.
  skipWhen: (** @type {State} */ state) =>
    !state.securityGroupId || !state[confirmDeleteSecurityGroup.name],
},
);

export const authorizeSecurityGroupIngress = new ScenarioAction(
  "authorizeSecurity",
  async (** @type {State} */ state) => {
    try {
      // Get the public IP address of the machine running this example.
      const ipAddress = await new Promise((res, rej) => {
        get("http://checkip.amazonaws.com", (response) => {
          let data = "";
          response.on("data", (chunk) => {
            data += chunk;
          });
          response.on("end", () => res(data.trim()));
        }).on("error", (err) => {
          rej(err);
        });
      });
      state.ipAddress = ipAddress;
      // Allow ingress from the IP address above to the security group.
      // This will allow you to SSH into the EC2 instance.
      const command = new AuthorizeSecurityGroupIngressCommand({
        GroupId: state.securityGroupId,
        IpPermissions: [
          {
            IpProtocol: "tcp",
            FromPort: 22,
            ToPort: 22,
            IpRanges: [{ CidrIp: `${ipAddress}/32` }],
          },
        ],
      });

      await state.ec2Client.send(command);
    }
  }
);
```



```
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidGroupId.Malformed"
      ) {
        caught.message = `${caught.message}. Please provide a valid GroupId.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const logSecurityGroupIngress = new ScenarioOutput(
  "logSecurityGroupIngress",
  (/** @type {State} */ state) =>
    `Allowed SSH access from your public IP: ${state.ipAddress}.`,
  { skipWhen: skipWhenErrors },
);

export const getImages = new ScenarioAction(
  "images",
  async (/** @type {State} */ state) => {
    const AMIs = [];
    // Some AWS services publish information about common artifacts as AWS
    Systems Manager (SSM)
    // public parameters. For example, the Amazon Elastic Compute Cloud (Amazon
    EC2)
    // service publishes information about Amazon Machine Images (AMIs) as public
    parameters.

    // Create the paginator for getting images. Actions that return multiple
    pages of
    // results have paginators to simplify those calls.
    const getParametersByPathPaginator = paginateGetParametersByPath(
      {
        // Not storing this client in state since it's only used once.
        client: new SSMClient({}),
      },
      {
        // The path to the public list of the latest amazon-linux instances.
        Path: "/aws/service/ami-amazon-linux-latest",
      },
    ),
```

```
);

try {
  for await (const page of getParametersByPathPaginator) {
    for (const param of page.Parameters) {
      // Filter by Amazon Linux 2
      if (param.Name.includes("amzn2")) {
        AMIs.push(param.Value);
      }
    }
  }
} catch (caught) {
  if (caught instanceof Error && caught.name === "InvalidFilterValue") {
    caught.message = `${caught.message} Please provide a valid filter value
for paginateGetParametersByPath.`;
  }
  state.errors.push(caught);
  return;
}

const imageDetails = [];
const describeImagesPaginator = paginateDescribeImages(
  { client: state.ec2Client },
  // The images found from the call to SSM.
  { ImageIds: AMIs },
);

try {
  // Get more details for the images found above.
  for await (const page of describeImagesPaginator) {
    imageDetails.push...(page.Images || []);
  }

  // Store the image details for later use.
  state.images = imageDetails;
} catch (caught) {
  if (caught instanceof Error && caught.name === "InvalidAMIID.NotFound") {
    caught.message = `${caught.message}. Please provide a valid image id.`;
  }

  state.errors.push(caught);
}
},
{ skipWhen: skipWhenErrors },
```

```
);

export const provideImage = new ScenarioInput(
  "image",
  "Select one of the following images.",
  {
    type: "select",
    choices: (/** @type { State } */ state) =>
      state.images.map((image) => ({
        name: `${image.Description}`,
        value: image,
      })),
    default: (/** @type { State } */ state) => state.images[0],
    skipWhen: skipWhenErrors,
  },
);

export const getCompatibleInstanceTypes = new ScenarioAction(
  "getCompatibleInstanceTypes",
  async (/** @type {State} */ state) => {
    // Get more details about instance types that match the architecture of
    // the provided image.
    const paginator = paginateDescribeInstanceTypes(
      { client: state.ec2Client, pageSize: 25 },
      {
        Filters: [
          {
            Name: "processor-info.supported-architecture",
            // The value selected from provideImage()
            Values: [state.image.Architecture],
          },
          // Filter for smaller, less expensive, types.
          { Name: "instance-type", Values: ["*.micro", "*.small"] },
        ],
      },
    );
  },
);

const instanceTypes = [];

try {
  for await (const page of paginator) {
    if (page.InstanceTypes.length) {
      instanceTypes.push(...(page.InstanceTypes || []));
    }
  }
}
```

```
    }

    if (!instanceTypes.length) {
      state.errors.push(
        "No instance types matched the instance type filters.",
      );
    }
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      caught.message = `${caught.message}. Please check the provided values and
      try again.`;
    }

    state.errors.push(caught);
  }

  state.instanceTypes = instanceTypes;
},
{ skipWhen: skipWhenErrors },
);

export const provideInstanceType = new ScenarioInput(
  "instanceType",
  "Select an instance type.",
  {
    choices: (/** @type {State} */ state) =>
      state.instanceTypes.map((instanceType) => ({
        name: `${instanceType.InstanceType} - Memory:
        ${instanceType.MemoryInfo.SizeInMiB}`,
        value: instanceType.InstanceType,
      })),
    type: "select",
    default: (/** @type {State} */ state) =>
      state.instanceTypes[0].InstanceType,
    skipWhen: skipWhenErrors,
  },
);

export const runInstance = new ScenarioAction(
  "runInstance",
  async (/** @type { State } */ state) => {
    const { Instances } = await state.ec2Client.send(
      new RunInstancesCommand({
        KeyName: state[provideKeyPairName.name],
```

```
    SecurityGroupIds: [state.securityGroupId],
    ImageId: state.image.ImageId,
    InstanceType: state[provideInstanceType.name],
    // Availability Zones have capacity limitations that may impact your
    // ability to launch instances.
    // The `RunInstances` operation will only succeed if it can allocate at
    // least the `MinCount` of instances.
    // However, EC2 will attempt to launch up to the `MaxCount` of instances,
    // even if the full request cannot be satisfied.
    // If you need a specific number of instances, use `MinCount` and
    // `MaxCount` set to the same value.
    // If you want to launch up to a certain number of instances, use
    // `MaxCount` and let EC2 provision as many as possible.
    // If you require a minimum number of instances, but do not want to
    // exceed a maximum, use both `MinCount` and `MaxCount`.
    MinCount: 1,
    MaxCount: 1,
  })),
);

state.instanceId = Instances[0].InstanceId;

try {
  // Poll `DescribeInstanceStatus` until status is "ok".
  await waitUntilInstanceStatusOk(
    {
      client: state.ec2Client,
      maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
    },
    { InstanceIds: [Instances[0].InstanceId] },
  );
} catch (caught) {
  if (caught instanceof Error && caught.name === "TimeoutError") {
    caught.message = `${caught.message}. Try increasing the maxWaitTime in
the waiter.`;
  }

  state.errors.push(caught);
}
},
{ skipWhen: skipWhenErrors },
);

export const logRunInstance = new ScenarioOutput(
```

```
"logRunInstance",
"The next step is to run your EC2 instance for the first time. This can take a
few minutes.",
{ header: true, skipWhen: skipWhenErrors },
);

export const describeInstance = new ScenarioAction(
  "describeInstance",
  async (** @type { State } */ state) => {
    /** @type { import("@aws-sdk/client-ec2").Instance[] } */
    const instances = [];

    try {
      const paginator = paginateDescribeInstances(
        {
          client: state.ec2Client,
        },
        {
          // Only get our created instance.
          InstanceIds: [state.instanceId],
        },
      );
    }

    for await (const page of paginator) {
      for (const reservation of page.Reservations) {
        instances.push(...reservation.Instances);
      }
    }
    if (instances.length !== 1) {
      throw new Error(`Instance ${state.instanceId} not found.`);
    }

    // The only info we need is the IP address for SSH purposes.
    state.instanceIpAddress = instances[0].PublicIpAddress;
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      caught.message = `${caught.message}. Please check provided values and try
again.`;
    }

    state.errors.push(caught);
  }
},
{ skipWhen: skipWhenErrors },
```

```
);

export const logSSHConnectionInfo = new ScenarioOutput(
  "logSSHConnectionInfo",
  (/** @type { State } */ state) =>
    `You can now SSH into your instance using the following command:
ssh -i ${state.tmpDirectory}/${state[provideKeyPairName.name]}.pem ec2-user@
${state.instanceIpAddress}`,
  { preformatted: true, skipWhen: skipWhenErrors },
);

export const logStopInstance = new ScenarioOutput(
  "logStopInstance",
  "Stopping your EC2 instance.",
  { skipWhen: skipWhenErrors },
);

export const stopInstance = new ScenarioAction(
  "stopInstance",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new StopInstancesCommand({
          InstanceIds: [state.instanceId],
        })),
    );

    await waitUntilInstanceStopped(
      {
        client: state.ec2Client,
        maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
      },
      { InstanceIds: [state.instanceId] },
    );
  } catch (caught) {
    if (caught instanceof Error && caught.name === "TimeoutError") {
      caught.message = `${caught.message}. Try increasing the maxWaitTime in
the waiter.`;
    }

    state.errors.push(caught);
  }
},
// Don't try to stop an instance that doesn't exist.
```

```
{ skipWhen: (/** @type { State } */ state) => !state.instanceId },
);

export const logIpAddressBehavior = new ScenarioOutput(
  "logIpAddressBehavior",
  [
    "When you run an instance, by default it's assigned an IP address.",
    "That IP address is not static. It will change every time the instance is
restarted.",
    "The next step is to stop and restart your instance to demonstrate this
behavior.",
  ].join(" "),
  { header: true, skipWhen: skipWhenErrors },
);

export const logStartInstance = new ScenarioOutput(
  "logStartInstance",
  (/** @type { State } */ state) => `Starting instance ${state.instanceId}`,
  { skipWhen: skipWhenErrors },
);

export const startInstance = new ScenarioAction(
  "startInstance",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new StartInstancesCommand({
          InstanceIds: [state.instanceId],
        }),
      );

      await waitUntilInstanceStatusOk(
        {
          client: state.ec2Client,
          maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
        },
        { InstanceIds: [state.instanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in
the waiter.`;
      }
    }
  }
);
```



```
        state.errors.push(caught);
    }
},
{ skipWhen: skipWhenErrors },
);

export const logIpAllocation = new ScenarioOutput(
    "logIpAllocation",
    [
        "It is possible to have a static IP address.",
        "To demonstrate this, an IP will be allocated and associated to your EC2
instance.",
    ].join(" "),
    { header: true, skipWhen: skipWhenErrors },
);

export const allocateIp = new ScenarioAction(
    "allocateIp",
    async (** @type { State } */ state) => {
        try {
            // An Elastic IP address is allocated to your AWS account, and is yours
            until you release it.
            const { AllocationId, PublicIp } = await state.ec2Client.send(
                new AllocateAddressCommand({}),
            );
            state.allocationId = AllocationId;
            state.allocatedIpAddress = PublicIp;
        } catch (caught) {
            if (caught instanceof Error && caught.name === "MissingParameter") {
                caught.message = `${caught.message}. Did you provide these values?`;
            }
            state.errors.push(caught);
        }
    },
    { skipWhen: skipWhenErrors },
);

export const associateIp = new ScenarioAction(
    "associateIp",
    async (** @type { State } */ state) => {
        try {
            // Associate an allocated IP address to an EC2 instance. An IP address can
            be allocated
            // with the AllocateAddress action.
```

```
const { AssociationId } = await state.ec2Client.send(
  new AssociateAddressCommand({
    AllocationId: state.allocationId,
    InstanceId: state.instanceId,
  }),
);
state.associationId = AssociationId;
// Update the IP address that is being tracked to match
// the one just associated.
state.instanceIpAddress = state.allocatedIpAddress;
} catch (caught) {
  if (
    caught instanceof Error &&
    caught.name === "InvalidAllocationID.NotFound"
  ) {
    caught.message = `${caught.message}. Did you provide the ID of a valid
Elastic IP address AllocationId?`;
  }
  state.errors.push(caught);
}
},
{ skipWhen: skipWhenErrors },
);

export const logStaticIpProof = new ScenarioOutput(
  "logStaticIpProof",
  "The IP address should remain the same even after stopping and starting the
instance.",
  { header: true, skipWhen: skipWhenErrors },
);

export const logCleanUp = new ScenarioOutput(
  "logCleanUp",
  "That's it! You can choose to clean up the resources now, or clean them up on
your own later.",
  { header: true, skipWhen: skipWhenErrors },
);

export const confirmDisassociateAddress = new ScenarioInput(
  "confirmDisassociateAddress",
  "Do you want to disassociate and release the static IP address created
earlier?",
  {
    type: "confirm",
  }
);
```

```
    skipWhen: (/** @type { State } */ state) => !state.associationId,
  },
);

export const maybeDisassociateAddress = new ScenarioAction(
  "maybeDisassociateAddress",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new DisassociateAddressCommand({
          AssociationId: state.associationId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidAssociationID.NotFound"
      ) {
        caught.message = `${caught.message}. Please provide a valid association
ID.`;
      }
      state.errors.push(caught);
    }
  },
  {
    skipWhen: (/** @type { State } */ state) =>
      !state[confirmDisassociateAddress.name] || !state.associationId,
  },
);

export const maybeReleaseAddress = new ScenarioAction(
  "maybeReleaseAddress",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new ReleaseAddressCommand({
          AllocationId: state.allocationId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidAllocationID.NotFound"
      ) {
```

```
        caught.message = `${caught.message}. Please provide a valid
AllocationID.`;
    }
    state.errors.push(caught);
  }
},
{
  skipWhen: (/** @type { State } */ state) =>
    !state[confirmDisassociateAddress.name] || !state.allocationId,
},
);

export const confirmTerminateInstance = new ScenarioInput(
  "confirmTerminateInstance",
  "Do you want to terminate the instance?",
  // Don't do anything when an instance was never run.
  {
    skipWhen: (/** @type { State } */ state) => !state.instanceId,
    type: "confirm",
  },
);

export const maybeTerminateInstance = new ScenarioAction(
  "terminateInstance",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new TerminateInstancesCommand({
          InstanceIds: [state.instanceId],
        }),
      );
      await waitUntilInstanceTerminated(
        { client: state.ec2Client },
        { InstanceIds: [state.instanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in
the waiter.`;
      }

      state.errors.push(caught);
    }
  },
),
```

```
{
  // Don't do anything when there's no instance to terminate or the
  // user chooses not to terminate.
  skipWhen: (** @type { State } */ state) =>
    !state.instanceId || !state[confirmTerminateInstance.name],
},
);

export const deleteTemporaryDirectory = new ScenarioAction(
  "deleteTemporaryDirectory",
  async (** @type { State } */ state) => {
    try {
      await rm(state.tmpDirectory, { recursive: true });
    } catch (caught) {
      state.errors.push(caught);
    }
  },
);

export const logErrors = new ScenarioOutput(
  "logErrors",
  (** @type {State}*/ state) => {
    const errorList = state.errors
      .map((err) => ` - ${err.name}: ${err.message}`)
      .join("\n");
    return `Scenario errors found:\n${errorList}`;
  },
  {
    preformatted: true,
    header: true,
    // Don't log errors when there aren't any!
    skipWhen: (** @type {State} */ state) => state.errors.length === 0,
  },
);
```

- API-Einheiten finden Sie in der AWS SDK for JavaScript API-Referenz zu den folgenden Themen.
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)

- [CreateSecurityGroup](#)
- [DeleteKeyPair](#)
- [DeleteSecurityGroup](#)
- [DescribeImages](#)
- [DescribeInstanceTypes](#)
- [DescribeInstances](#)
- [DescribeKeyPairs](#)
- [DescribeSecurityGroups](#)
- [DisassociateAddress](#)
- [ReleaseAddress](#)
- [RunInstances](#)
- [StartInstances](#)
- [StopInstances](#)
- [TerminateInstances](#)
- [UnmonitorInstances](#)

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
```

```
Before running this Kotlin 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-kotlin/latest/developer-guide/setup.html
```

1. Creates an RSA key pair and saves the private key data as a .pem file.
 2. Lists key pairs.
 3. Creates a security group for the default VPC.
 4. Displays security group information.
 5. Gets a list of Amazon Linux 2 AMIs and selects one.
 6. Gets more information about the image.
 7. Gets a list of instance types that are compatible with the selected AMI's architecture.
 8. Creates an instance with the key pair, security group, AMI, and an instance type.
 9. Displays information about the instance.
 10. Stops the instance and waits for it to stop.
 11. Starts the instance and waits for it to start.
 12. Allocates an Elastic IP address and associates it with the instance.
 13. Displays SSH connection info for the instance.
 14. Disassociates and deletes the Elastic IP address.
 15. Terminates the instance.
 16. Deletes the security group.
 17. Deletes the key pair.
- */

```
val DASHES = String(CharArray(80)).replace("\u0000", "-")
```

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <keyName> <fileName> <groupName> <groupDesc> <vpcId> <myIpAddress>

        Where:
            keyName - A key pair name (for example, TestKeyPair).
            fileName - A file name where the key information is written to.
            groupName - The name of the security group.
            groupDesc - The description of the security group.
            vpcId - A VPC ID. You can get this value from the AWS Management
Console.
            myIpAddress - The IP address of your development machine.

        """

    if (args.size != 6) {
        println(usage)
        exitProcess(0)
    }
}
```

```
val keyName = args[0]
val fileName = args[1]
val groupName = args[2]
val groupDesc = args[3]
val vpcId = args[4]
val myIpAddress = args[5]
var newInstanceId: String? = ""

println(DASHES)
println("Welcome to the Amazon EC2 example scenario.")
println(DASHES)

println(DASHES)
println("1. Create an RSA key pair and save the private key material as
a .pem file.")
createKeyPairSc(keyName, fileName)
println(DASHES)

println(DASHES)
println("2. List key pairs.")
describeEC2KeysSc()
println(DASHES)

println(DASHES)
println("3. Create a security group.")
val groupId = createEC2SecurityGroupSc(groupName, groupDesc, vpcId,
myIpAddress)
println(DASHES)

println(DASHES)
println("4. Display security group info for the newly created security
group.")
describeSecurityGroupsSc(groupId.toString())
println(DASHES)

println(DASHES)
println("5. Get a list of Amazon Linux 2 AMIs and select one with amzn2 in
the name.")
val instanceId = getParaValuesSc()
if (instanceId == "") {
    println("The instance Id value isn't valid.")
    exitProcess(0)
}
```



```
println("The instance Id is $instanceId.")
println(DASHES)

println(DASHES)
println("6. Get more information about an amzn2 image and return the AMI
value.")
val amiValue = instanceId?.let { describeImageSc(it) }
if (instanceId == "") {
    println("The instance Id value is invalid.")
    exitProcess(0)
}
println("The AMI value is $amiValue.")
println(DASHES)

println(DASHES)
println("7. Get a list of instance types.")
val instanceType = getInstanceTypesSc()
println(DASHES)

println(DASHES)
println("8. Create an instance.")
if (amiValue != null) {
    newInstanceId = runInstanceSc(instanceType, keyName, groupName, amiValue)
    println("The instance Id is $newInstanceId")
}
println(DASHES)

println(DASHES)
println("9. Display information about the running instance. ")
var ipAddress = describeEC2InstancesSc(newInstanceId)
println("You can SSH to the instance using this command:")
println("ssh -i " + fileName + "ec2-user@" + ipAddress)
println(DASHES)

println(DASHES)
println("10. Stop the instance.")
if (newInstanceId != null) {
    stopInstanceSc(newInstanceId)
}
println(DASHES)

println(DASHES)
println("11. Start the instance.")
if (newInstanceId != null) {
```

```
        startInstanceSc(newInstanceId)
    }
    ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("12. Allocate an Elastic IP address and associate it with the
instance.")
    val allocationId = allocateAddressSc()
    println("The allocation Id value is $allocationId")
    val associationId = associateAddressSc(newInstanceId, allocationId)
    println("The associate Id value is $associationId")
    println(DASHES)

    println(DASHES)
    println("13. Describe the instance again.")
    ipAddress = describeEC2InstancesSc(newInstanceId)
    println("You can SSH to the instance using this command:")
    println("ssh -i " + fileName + "ec2-user@" + ipAddress)
    println(DASHES)

    println(DASHES)
    println("14. Disassociate and release the Elastic IP address.")
    disassociateAddressSc(associationId)
    releaseEC2AddressSc(allocationId)
    println(DASHES)

    println(DASHES)
    println("15. Terminate the instance and use a waiter.")
    if (newInstanceId != null) {
        terminateEC2Sc(newInstanceId)
    }
    println(DASHES)

    println(DASHES)
    println("16. Delete the security group.")
    if (groupId != null) {
        deleteEC2SecGroupSc(groupId)
    }
    println(DASHES)

    println(DASHES)
```

```
println("17. Delete the key pair.")
deleteKeysSc(keyName)
println(DASHES)

println(DASHES)
println("You successfully completed the Amazon EC2 scenario.")
println(DASHES)
}

suspend fun deleteKeysSc(keyPair: String) {
    val request =
        DeleteKeyPairRequest {
            keyName = keyPair
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteKeyPair(request)
        println("Successfully deleted key pair named $keyPair")
    }
}

suspend fun deleteEC2SecGroupSc(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted security group with Id $groupIdVal")
    }
}

suspend fun terminateEC2Sc(instanceIdVal: String) {
    val ti =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceIdVal)
        }
    println("Wait for the instance to terminate. This will take a few minutes.")
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.terminateInstances(ti)
        ec2.waitForInstanceTerminated {
            // suspend call
            instanceIds = listOf(instanceIdVal)
        }
        println("$instanceIdVal is terminated!")
    }
}
```

```
    }
}

suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.releaseAddress(request)
        println("Successfully released Elastic IP address $allocId")
    }
}

suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}

suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}

suspend fun allocateAddressSc(): String? {
    val allocateRequest =
```

```
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
        return allocateResponse.allocationId
    }
}

suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
minutes.")
        ec2.waitForInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}

suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
minutes.")
        ec2.waitForInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}
```

```
suspend fun describeEC2InstancesSc(newInstanceId: String?): String {
    var pubAddress = ""
    var isRunning = false
    val request =
        DescribeInstancesRequest {
            instanceIds = listOf(newInstanceId.toString())
        }

    while (!isRunning) {
        Ec2Client { region = "us-west-2" }.use { ec2 ->
            val response = ec2.describeInstances(request)
            val state =
                response.reservations
                    ?.get(0)
                    ?.instances
                    ?.get(0)
                    ?.state
                    ?.name
                    ?.value
            if (state != null) {
                if (state.compareTo("running") == 0) {
                    println("Image id is
${response.reservations!!.get(0).instances?.get(0)?.imageId}")
                    println("Instance type is
${response.reservations!!.get(0).instances?.get(0)?.instanceType}")
                    println("Instance state is
${response.reservations!!.get(0).instances?.get(0)?.state}")
                    pubAddress =
                        response.reservations!!
                            .get(0)
                            .instances
                            ?.get(0)
                            ?.publicIpAddress
                            .toString()
                    println("Instance address is $pubAddress")
                    isRunning = true
                }
            }
        }
    }
    return pubAddress
}
```

```
suspend fun runInstanceSc(
    instanceTypeVal: String,
    keyNameVal: String,
    groupNameVal: String,
    amiIdVal: String,
): String {
    val runRequest =
        RunInstancesRequest {
            instanceType = InstanceType.fromValue(instanceTypeVal)
            keyName = keyNameVal
            securityGroups = listOf(groupNameVal)
            maxCount = 1
            minCount = 1
            imageId = amiIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(runRequest)
        val instanceId = response.instances?.get(0)?.instanceId
        println("Successfully started EC2 Instance $instanceId based on AMI
$amiIdVal")
        return instanceId.toString()
    }
}

// Get a list of instance types.
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
```

```
        println("The memory information of this type is
${type.memoryInfo?.sizeInMib}")
        println("Maximum number of network cards is
${type.networkInfo?.maximumNetworkCards}")
        instanceType = type.instanceType.toString()
    }
    return instanceType
}
}

// Display the Description field that corresponds to the instance Id value.
suspend fun describeImageSc(instanceId: String): String? {
    val imagesRequest =
        DescribeImagesRequest {
            imageIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeImages(imagesRequest)
        println("The description of the first image is
${response.images?.get(0)?.description}")
        println("The name of the first image is
${response.images?.get(0)?.name}")

        // Return the image Id value.
        return response.images?.get(0)?.imageId
    }
}

// Get the Id value of an instance with amzn2 in the name.
suspend fun getParaValuesSc(): String? {
    val parameterRequest =
        GetParametersByPathRequest {
            path = "/aws/service/ami-amazon-linux-latest"
        }

    SsmClient { region = "us-west-2" }.use { ssmClient ->
        val response = ssmClient.getParametersByPath(parameterRequest)
        response.parameters?.forEach { para ->
            println("The name of the para is: ${para.name}")
            println("The type of the para is: ${para.type}")
            println("")
            if (para.name?.let { filterName(it) } == true) {
                return para.value
            }
        }
    }
}
```



```
    }
  }
}
return ""
}

fun filterName(name: String): Boolean {
    val parts = name.split("/").toTypedArray()
    val myValue = parts[4]
    return myValue.contains("amzn2")
}

suspend fun describeSecurityGroupsSc(groupId: String) {
    val request =
        DescribeSecurityGroupsRequest {
            groupIds = listOf(groupId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeSecurityGroups(request)
        for (group in response.securityGroups!!) {
            println("Found Security Group with id " + group.groupId.toString() +
" and group VPC " + group.vpcId)
        }
    }
}

suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
```

```
        cidrIp = "$myIpAddress/0"
    }

    val ipPerm =
        IpPermission {
            ipProtocol = "tcp"
            toPort = 80
            fromPort = 80
            ipRanges = listOf(ipRange)
        }

    val ipPerm2 =
        IpPermission {
            ipProtocol = "tcp"
            toPort = 22
            fromPort = 22
            ipRanges = listOf(ipRange)
        }

    val authRequest =
        AuthorizeSecurityGroupIngressRequest {
            groupName = groupNameVal
            ipPermissions = listOf(ipPerm, ipPerm2)
        }
    ec2.authorizeSecurityGroupIngress(authRequest)
    println("Successfully added ingress policy to Security Group
    $groupNameVal")
    return resp.groupId
}

suspend fun describeEC2KeysSc() {
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
            ${keyPair.keyFingerprint}")
        }
    }
}

suspend fun createKeyPairSc(
    keyNameVal: String,
    fileNameVal: String,
```

```
) {  
    val request =  
        CreateKeyPairRequest {  
            keyName = keyNameVal  
        }  
  
    Ec2Client { region = "us-west-2" }.use { ec2 ->  
        val response = ec2.createKeyPair(request)  
        val content = response.keyMaterial  
        if (content != null) {  
            File(fileNameVal).writeText(content)  
        }  
        println("Successfully created key pair named $keyNameVal")  
    }  
}
```

- APIEinzelheiten finden Sie in AWS SDK den folgenden Themen als APIKotlin-Referenz.
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)
 - [DeleteSecurityGroup](#)
 - [DescribeImages](#)
 - [DescribeInstanceTypes](#)
 - [DescribeInstances](#)
 - [DescribeKeyPairs](#)
 - [DescribeSecurityGroups](#)
 - [DisassociateAddress](#)
 - [ReleaseAddress](#)
 - [RunInstances](#)
 - [StartInstances](#)
 - [StopInstances](#)
 - [TerminateInstances](#)

- [UnmonitorInstances](#)

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
class EC2InstanceScenario:
    """
    A scenario that demonstrates how to use Boto3 to manage Amazon EC2 resources.
    Covers creating a key pair, security group, launching an instance,
    associating
    an Elastic IP, and cleaning up resources.
    """

    def __init__(
        self,
        inst_wrapper: EC2InstanceWrapper,
        key_wrapper: KeyPairWrapper,
        sg_wrapper: SecurityGroupWrapper,
        eip_wrapper: ElasticIpWrapper,
        ssm_client: boto3.client,
        remote_exec: bool = False,
    ):
        """
        Initializes the EC2InstanceScenario with the necessary AWS service
        wrappers.

        :param inst_wrapper: Wrapper for EC2 instance operations.
        :param key_wrapper: Wrapper for key pair operations.
        :param sg_wrapper: Wrapper for security group operations.
        :param eip_wrapper: Wrapper for Elastic IP operations.
        :param ssm_client: Boto3 client for accessing SSM to retrieve AMIs.
        :param remote_exec: Flag to indicate if the scenario is running in a
        remote execution
```

```
environment. Defaults to False. If True, the script
won't prompt
        for user interaction.
    """
    self.inst_wrapper = inst_wrapper
    self.key_wrapper = key_wrapper
    self.sg_wrapper = sg_wrapper
    self.eip_wrapper = eip_wrapper
    self.ssm_client = ssm_client
    self.remote_exec = remote_exec

    def create_and_list_key_pairs(self) -> None:
        """
        Creates an RSA key pair for SSH access to the EC2 instance and lists
        available key pairs.
        """
        console.print("***Step 1: Create a Secure Key Pair***", style="bold cyan")
        console.print(
            "Let's create a secure RSA key pair for connecting to your EC2
            instance."
        )
        key_name = f"MyUniqueKeyPair-{uuid.uuid4().hex[:8]}"
        console.print(f"- **Key Pair Name**: {key_name}")

        # Create the key pair and simulate the process with a progress bar.
        with alive_bar(1, title="Creating Key Pair") as bar:
            self.key_wrapper.create(key_name)
            time.sleep(0.4) # Simulate the delay in key creation
            bar()

        console.print(f"- **Private Key Saved to**:"
{self.key_wrapper.key_file_path}\n")

        # List key pairs (simulated) and show a progress bar.
        list_keys = True
        if list_keys:
            console.print("- Listing your key pairs...")
            start_time = time.time()
            with alive_bar(100, title="Listing Key Pairs") as bar:
                while time.time() - start_time < 2:
                    time.sleep(0.2)
                    bar(10)
                self.key_wrapper.list(5)
            if time.time() - start_time > 2:
```

```
        console.print(
            "Taking longer than expected! Please wait...",
            style="bold yellow",
        )

def create_security_group(self) -> None:
    """
    Creates a security group that controls access to the EC2 instance and
adds a rule
to allow SSH access from the user's current public IP address.
    """
    console.print("***Step 2: Create a Security Group***", style="bold cyan")
    console.print(
        "Security groups manage access to your instance. Let's create one."
    )
    sg_name = f"MySecurityGroup-{uuid.uuid4().hex[:8]}"
    console.print(f"- **Security Group Name**: {sg_name}")

    # Create the security group and simulate the process with a progress bar.
with alive_bar(1, title="Creating Security Group") as bar:
        self.sg_wrapper.create(
            sg_name, "Security group for example: get started with
instances."
        )
        time.sleep(0.5)
        bar()

    console.print(f"- **Security Group ID**:
{self.sg_wrapper.security_group}\n")

    # Get the current public IP to set up SSH access.
ip_response = urllib.request.urlopen("http://checkip.amazonaws.com")
current_ip_address = ip_response.read().decode("utf-8").strip()
    console.print(
        "Let's add a rule to allow SSH only from your current IP address."
    )
    console.print(f"- **Your Public IP Address**: {current_ip_address}")
    console.print("- Automatically adding SSH rule...")

    # Update security group rules to allow SSH and simulate with a progress
bar.
with alive_bar(1, title="Updating Security Group Rules") as bar:
        response = self.sg_wrapper.authorize_ingress(current_ip_address)
        time.sleep(0.4)
```

```
        if response and response.get("Return"):
            console.print("- **Security Group Rules Updated**.")
        else:
            console.print(
                "- **Error**: Couldn't update security group rules.",
                style="bold red",
            )
        bar()

    self.sg_wrapper.describe(self.sg_wrapper.security_group)

def create_instance(self) -> None:
    """
    Launches an EC2 instance using an Amazon Linux 2 AMI and the created key
pair
and security group. Displays instance details and SSH connection
information.
    """
    # Retrieve Amazon Linux 2 AMIs from SSM.
    ami_paginator = self.ssm_client.get_paginator("get_parameters_by_path")
    ami_options = []
    for page in ami_paginator.paginate(Path="/aws/service/ami-amazon-linux-
latest"):
        ami_options += page["Parameters"]
    amzn2_images = self.inst_wrapper.get_images(
        [opt["Value"] for opt in ami_options if "amzn2" in opt["Name"]]
    )
    console.print("\n**Step 3: Launch Your Instance**", style="bold cyan")
    console.print(
        "Let's create an instance from an Amazon Linux 2 AMI. Here are some
options:"
    )
    image_choice = 0
    console.print(f"- Selected AMI: {amzn2_images[image_choice]
['ImageId']}\n")

    # Display instance types compatible with the selected AMI
    inst_types = self.inst_wrapper.get_instance_types(
        amzn2_images[image_choice]["Architecture"]
    )
    inst_type_choice = 0
    console.print(
        f"- Selected instance type: {inst_types[inst_type_choice]
['InstanceType']}\n"
```

```
)

console.print("Creating your instance and waiting for it to start...")
with alive_bar(1, title="Creating Instance") as bar:
    self.inst_wrapper.create(
        amzn2_images[image_choice]["ImageId"],
        inst_types[inst_type_choice]["InstanceType"],
        self.key_wrapper.key_pair["KeyName"],
        [self.sg_wrapper.security_group],
    )
    time.sleep(21)
    bar()

console.print(f"***Success! Your instance is ready:**\n", style="bold
green")
self.inst_wrapper.display()

console.print(
    "You can use SSH to connect to your instance. "
    "If the connection attempt times out, you might have to manually
update "
    "the SSH ingress rule for your IP address in the AWS Management
Console."
)
self._display_ssh_info()

def _display_ssh_info(self) -> None:
    """
    Displays SSH connection information for the user to connect to the EC2
instance.
    Handles the case where the instance does or does not have an associated
public IP address.
    """
    if (
        not self.eip_wrapper.elastic_ips
        or not self.eip_wrapper.elastic_ips[0].allocation_id
    ):
        if self.inst_wrapper.instances:
            instance = self.inst_wrapper.instances[0]
            instance_id = instance["InstanceId"]

            waiter =
self.inst_wrapper.ec2_client.get_waiter("instance_running")
            console.print(
```



```
        "Waiting for the instance to be in a running state with a
public IP...",
        style="bold cyan",
    )

    with alive_bar(1, title="Waiting for Instance to Start") as bar:
        waiter.wait(InstanceIds=[instance_id])
        time.sleep(20)
        bar()

    instance = self.inst_wrapper.ec2_client.describe_instances(
        InstanceIds=[instance_id]
    )["Reservations"][0]["Instances"][0]

    public_ip = instance.get("PublicIpAddress")
    if public_ip:
        console.print(
            "\nTo connect via SSH, open another command prompt and
run the following command:",
            style="bold cyan",
        )
        console.print(
            f"\tssh -i {self.key_wrapper.key_file_path} ec2-
user@{public_ip}"
        )
    else:
        console.print(
            "Instance does not have a public IP address assigned.",
            style="bold red",
        )
    else:
        console.print(
            "No instance available to retrieve public IP address.",
            style="bold red",
        )
    else:
        elastic_ip = self.eip_wrapper.elastic_ips[0]
        elastic_ip_address = elastic_ip.public_ip
        console.print(
            f"\tssh -i {self.key_wrapper.key_file_path} ec2-
user@{elastic_ip_address}"
        )

    if not self.remote_exec:
```

```
        console.print("\nOpen a new terminal tab to try the above SSH
command.")
        input("Press Enter to continue...")

    def associate_elastic_ip(self) -> None:
        """
        Allocates an Elastic IP address and associates it with the EC2 instance.
        Displays the Elastic IP address and SSH connection information.
        """
        console.print("\n**Step 4: Allocate an Elastic IP Address**", style="bold
cyan")
        console.print(
            "You can allocate an Elastic IP address and associate it with your
instance\n"
            "to keep a consistent IP address even when your instance restarts."
        )

        with alive_bar(1, title="Allocating Elastic IP") as bar:
            elastic_ip = self.eip_wrapper.allocate()
            time.sleep(0.5)
            bar()

        console.print(
            f"- **Allocated Static Elastic IP Address**: {elastic_ip.public_ip}."
        )

        with alive_bar(1, title="Associating Elastic IP") as bar:
            self.eip_wrapper.associate(
                elastic_ip.allocation_id, self.inst_wrapper.instances[0]
["InstanceId"]
            )
            time.sleep(2)
            bar()

        console.print(f"- **Associated Elastic IP with Your Instance**.")
        console.print(
            "You can now use SSH to connect to your instance by using the Elastic
IP."
        )
        self._display_ssh_info()

    def stop_and_start_instance(self) -> None:
        """
        Stops and restarts the EC2 instance. Displays instance state and explains
```

```
changes that occur when the instance is restarted, such as the potential
change
in the public IP address unless an Elastic IP is associated.
"""
console.print("\n**Step 5: Stop and Start Your Instance**", style="bold
cyan")
console.print("Let's stop and start your instance to see what changes.")
console.print("- **Stopping your instance and waiting until it's
stopped...**")

with alive_bar(1, title="Stopping Instance") as bar:
    self.inst_wrapper.stop()
    time.sleep(360)
    bar()

console.print("- **Your instance is stopped. Restarting...**")

with alive_bar(1, title="Starting Instance") as bar:
    self.inst_wrapper.start()
    time.sleep(20)
    bar()

console.print("**Your instance is running.**", style="bold green")
self.inst_wrapper.display()

elastic_ip = (
    self.eip_wrapper.elastic_ips[0] if self.eip_wrapper.elastic_ips else
None
)

if elastic_ip is None or elastic_ip.allocation_id is None:
    console.print(
        "- **Note**: Every time your instance is restarted, its public IP
address changes."
    )
else:
    console.print(
        f"Because you have associated an Elastic IP with your instance,
you can \n"
        f"connect by using a consistent IP address after the instance
restarts: {elastic_ip.public_ip}"
    )

self._display_ssh_info()
```

```
def cleanup(self) -> None:
    """
    Cleans up all the resources created during the scenario, including
    disassociating
    and releasing the Elastic IP, terminating the instance, deleting the
    security
    group, and deleting the key pair.
    """
    console.print("\n**Step 6: Clean Up Resources**", style="bold cyan")
    console.print("Cleaning up resources:")

    for elastic_ip in self.eip_wrapper.elastic_ips:
        console.print(f"- **Elastic IP**: {elastic_ip.public_ip}")

        with alive_bar(1, title="Disassociating Elastic IP") as bar:
            self.eip_wrapper.disassociate(elastic_ip.allocation_id)
            time.sleep(2)
            bar()

        console.print("\t- **Disassociated Elastic IP from the Instance**")

        with alive_bar(1, title="Releasing Elastic IP") as bar:
            self.eip_wrapper.release(elastic_ip.allocation_id)
            time.sleep(1)
            bar()

        console.print("\t- **Released Elastic IP**")

    console.print(f"- **Instance**: {self.inst_wrapper.instances[0]
['InstanceId']}")

    with alive_bar(1, title="Terminating Instance") as bar:
        self.inst_wrapper.terminate()
        time.sleep(380)
        bar()

    console.print("\t- **Terminated Instance**")

    console.print(f"- **Security Group**: {self.sg_wrapper.security_group}")

    with alive_bar(1, title="Deleting Security Group") as bar:
        self.sg_wrapper.delete(self.sg_wrapper.security_group)
        time.sleep(1)
```

```
        bar()

        console.print("\t- **Deleted Security Group**")

        console.print(f"- **Key Pair**: {self.key_wrapper.key_pair['KeyName']}")

        with alive_bar(1, title="Deleting Key Pair") as bar:
            self.key_wrapper.delete(self.key_wrapper.key_pair["KeyName"])
            time.sleep(0.4)
            bar()

        console.print("\t- **Deleted Key Pair**")

    def run_scenario(self) -> None:
        """
        Executes the entire EC2 instance scenario: creates key pairs, security
        groups,
        launches an instance, associates an Elastic IP, and cleans up all
        resources.
        """
        logging.basicConfig(level=logging.INFO, format="%(levelname)s:
%(message)s")

        console.print("-" * 88)
        console.print(
            "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started
            with instances demo.",
            style="bold magenta",
        )
        console.print("-" * 88)

        self.create_and_list_key_pairs()
        self.create_security_group()
        self.create_instance()
        self.stop_and_start_instance()
        self.associate_elastic_ip()
        self.stop_and_start_instance()
        self.cleanup()

        console.print("\nThanks for watching!", style="bold green")
        console.print("-" * 88)

if __name__ == "__main__":
```

```

try:
    scenario = EC2InstanceScenario(
        EC2InstanceWrapper.from_client(),
        KeyPairWrapper.from_client(),
        SecurityGroupWrapper.from_client(),
        ElasticIpWrapper.from_client(),
        boto3.client("ssm"),
    )
    scenario.run_scenario()
except Exception:
    logging.exception("Something went wrong with the demo.")

```

Definieren Sie eine Klasse, die Schlüsselpaar-Aktionen umschließt.

```

class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
        directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is
        stored.
                           This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                           This is a high-level object that wraps key pair actions.
        Optional.
        """
        self.ec2_client = ec2_client

```

```
self.key_pair = key_pair
self.key_file_path: Optional[str] = None
self.key_file_dir = key_file_dir

@classmethod
def from_client(cls) -> "KeyPairWrapper":
    """
    Class method to create an instance of KeyPairWrapper using a new EC2
client
and a temporary directory for storing key files.

:return: An instance of KeyPairWrapper.
    """
    ec2_client = boto3.client("ec2")
    return cls(ec2_client, tempfile.TemporaryDirectory())

def create(self, key_name: str) -> dict:
    """
    Creates a key pair that can be used to securely connect to an EC2
instance.
    The returned key pair contains private key information that cannot be
retrieved
again. The private key data is stored as a .pem file.

:param key_name: The name of the key pair to create.
:return: A dictionary representing the Boto3 KeyPair object that
represents the newly created key pair.
:raises ClientError: If there is an error in creating the key pair, for
example, if a key pair with the same name already exists.
    """
    try:
        response = self.ec2_client.create_key_pair(KeyName=key_name)
        self.key_pair = response
        self.key_file_path = os.path.join(
            self.key_file_dir.name, f"{self.key_pair['KeyName']}.pem"
        )
        with open(self.key_file_path, "w") as key_file:
            key_file.write(self.key_pair["KeyMaterial"])
    except ClientError as err:
        if err.response["Error"]["Code"] == "InvalidKeyPair.Duplicate":
            logger.error(
                f"A key pair called {key_name} already exists. "
                "Please choose a different name for your key pair "
```

```
        "or delete the existing key pair before creating."
    )
    raise
else:
    return self.key_pair

def list(self, limit: Optional[int] = None) -> None:
    """
    Displays a list of key pairs for the current account.

    WARNING: Results are not paginated.

    :param limit: The maximum number of key pairs to list. If not specified,
                  all key pairs will be listed.
    :raises ClientError: If there is an error in listing the key pairs.
    """
    try:
        response = self.ec2_client.describe_key_pairs()
        key_pairs = response.get("KeyPairs", [])

        if limit:
            key_pairs = key_pairs[:limit]

        for key_pair in key_pairs:
            logger.info(
                f"Found {key_pair['KeyType']} key '{key_pair['KeyName']}'
with fingerprint:"
            )
            logger.info(f"\t{key_pair['KeyFingerprint']}")
    except ClientError as err:
        logger.error(f"Failed to list key pairs: {str(err)}")
        raise

def delete(self, key_name: str) -> bool:
    """
    Deletes a key pair by its name.

    :param key_name: The name of the key pair to delete.
    :return: A boolean indicating whether the deletion was successful.
    :raises ClientError: If there is an error in deleting the key pair, for
example,
                    if the key pair does not exist.
```



```

    """
    try:
        self.ec2_client.delete_key_pair(KeyName=key_name)
        logger.info(f"Successfully deleted key pair: {key_name}")
        self.key_pair = None
        return True
    except self.ec2_client.exceptions.ClientError as err:
        logger.error(f"Deletion failed for key pair: {key_name}")
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidKeyPair.NotFound":
            logger.error(
                f"The key pair '{key_name}' does not exist and cannot be
deleted. "
                "Please verify the key pair name and try again."
            )
            raise

```

Definieren Sie eine Klasse, die Sicherheitsgruppen-Aktionen umschließt.

```

class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                        access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
high-level identifier
                                that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

```

```
@classmethod
def from_client(cls) -> "SecurityGroupWrapper":
    """
    Creates a SecurityGroupWrapper instance with a default EC2 client.

    :return: An instance of SecurityGroupWrapper initialized with the default
    EC2 client.
    """
    ec2_client = boto3.client("ec2")
    return cls(ec2_client)

def create(self, group_name: str, group_description: str) -> str:
    """
    Creates a security group in the default virtual private cloud (VPC) of
    the current account.

    :param group_name: The name of the security group to create.
    :param group_description: The description of the security group to
    create.
    :return: The ID of the newly created security group.
    :raise Handles AWS SDK service-level ClientError, with special handling
    for ResourceAlreadyExists
    """
    try:
        response = self.ec2_client.create_security_group(
            GroupName=group_name, Description=group_description
        )
        self.security_group = response["GroupId"]
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceAlreadyExists":
            logger.error(
                f"Security group '{group_name}' already exists. Please choose
                a different name."
            )
            raise
        else:
            return self.security_group

def authorize_ingress(self, ssh_ingress_ip: str) -> Optional[Dict[str, Any]]:
    """
    Adds a rule to the security group to allow access to SSH.
```

```

        :param ssh_ingress_ip: The IP address that is granted inbound access to
connect
                                to port 22 over TCP, used for SSH.
        :return: The response to the authorization request. The 'Return' field of
the
                response indicates whether the request succeeded or failed, or
None if no security group is set.
        :raise Handles AWS SDK service-level ClientError, with special handling
for ResourceAlreadyExists
        """
        if self.security_group is None:
            logger.info("No security group to update.")
            return None

        try:
            ip_permissions = [
                {
                    # SSH ingress open to only the specified IP address.
                    "IpProtocol": "tcp",
                    "FromPort": 22,
                    "ToPort": 22,
                    "IpRanges": [{"CidrIp": f"{ssh_ingress_ip}/32"}],
                }
            ]
            response = self.ec2_client.authorize_security_group_ingress(
                GroupId=self.security_group, IpPermissions=ip_permissions
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidPermission.Duplicate":
                logger.error(
                    f"The SSH ingress rule for IP {ssh_ingress_ip} already
exists"
                    f"in security group '{self.security_group}'."
                )
                raise
            else:
                return response

    def describe(self, security_group_id: Optional[str] = None) -> bool:
        """
        Displays information about the specified security group or all security
groups if no ID is provided.

```

```

    :param security_group_id: The ID of the security group to describe.
                               If None, an open search is performed to
describe all security groups.
    :returns: True if the description is successful.
    :raises ClientError: If there is an error describing the security
group(s), such as an invalid security group ID.
    """
    try:
        paginator = self.ec2_client.get_paginator("describe_security_groups")

        if security_group_id is None:
            # If no ID is provided, return all security groups.
            page_iterator = paginator.paginate()
        else:
            page_iterator = paginator.paginate(GroupIds=[security_group_id])

        for page in page_iterator:
            for security_group in page["SecurityGroups"]:
                print(f"Security group: {security_group['GroupName']}")
                print(f"\tID: {security_group['GroupId']}")
                print(f"\tVPC: {security_group['VpcId']}")
                if security_group["IpPermissions"]:
                    print("Inbound permissions:")
                    pp(security_group["IpPermissions"])

        return True
    except ClientError as err:
        logger.error("Failed to describe security group(s).")
        if err.response["Error"]["Code"] == "InvalidGroup.NotFound":
            logger.error(
                f"Security group {security_group_id} does not exist "
                f"because the specified security group ID was not found."
            )
        raise

def delete(self, security_group_id: str) -> bool:
    """
    Deletes the specified security group.

    :param security_group_id: The ID of the security group to delete.
Required.

    :returns: True if the deletion is successful.

```

```

        :raises ClientError: If the security group cannot be deleted due to an
AWS service error.
        """
        try:
            self.ec2_client.delete_security_group(GroupId=security_group_id)
            logger.info(f"Successfully deleted security group
'{security_group_id}'")
            return True
        except ClientError as err:
            logger.error(f"Deletion failed for security group
'{security_group_id}'")
            error_code = err.response["Error"]["Code"]

            if error_code == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted
because it does not exist."
                )
            elif error_code == "DependencyViolation":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted
because it is still in use."
                    "\n\t- Detached from resources"
                    "\n\t- Removed from references in other groups"
                    "\n\t- Removed from VPC's as a default group"
                )
            raise

```

Definieren Sie eine Klasse, die Instance-Aktionen umschließt.

```

class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """

```

```
    Initializes the EC2InstanceWrapper with an EC2 client and optional
instances.

    :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                        access to AWS EC2 services.
    :param instances: A list of dictionaries representing Boto3 Instance
objects. These are high-level objects that
                        wrap instance actions.
    """
    self.ec2_client = ec2_client
    self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def create(
        self,
        image_id: str,
        instance_type: str,
        key_pair_name: str,
        security_group_ids: Optional[List[str]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Creates a new EC2 instance in the default VPC of the current account.

        The instance starts immediately after it is created.

        :param image_id: The ID of the Amazon Machine Image (AMI) to use for the
instance.
        :param instance_type: The type of instance to create, such as 't2.micro'.
        :param key_pair_name: The name of the key pair to use for SSH access.
        :param security_group_ids: A list of security group IDs to associate with
the instance.
```

```

                                If not specified, the default security group
of the VPC is used.
    :return: A list of dictionaries representing Boto3 Instance objects
representing the newly created instances.
    """
    try:
        instance_params = {
            "ImageId": image_id,
            "InstanceType": instance_type,
            "KeyName": key_pair_name,
        }
        if security_group_ids is not None:
            instance_params["SecurityGroupIds"] = security_group_ids

        response = self.ec2_client.run_instances(
            **instance_params, MinCount=1, MaxCount=1
        )
        instance = response["Instances"][0]
        self.instances.append(instance)
        waiter = self.ec2_client.get_waiter("instance_running")
        waiter.wait(InstanceIds=[instance["InstanceId"]])
    except ClientError as err:
        params_str = "\n\t".join(
            f"{key}: {value}" for key, value in instance_params.items()
        )
        logger.error(
            f"Failed to complete instance creation request.\nRequest details:
{params_str}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "InstanceLimitExceeded":
            logger.error(
                (
                    f"Insufficient capacity for instance type
'{{instance_type}}'. "
                    "Terminate unused instances or contact AWS Support for a
limit increase."
                )
            )
        if error_code == "InsufficientInstanceCapacity":
            logger.error(
                (
                    f"Insufficient capacity for instance type
'{{instance_type}}'. "
```

```

        "Select a different instance type or launch in a
different availability zone."
    )
    )
    raise
    return self.instances

def display(self, state_filter: Optional[str] = "running") -> None:
    """
    Displays information about instances, filtering by the specified state.

    :param state_filter: The instance state to include in the output. Only
instances in this state
        will be displayed. Default is 'running'. Example
states: 'running', 'stopped'.
    """
    if not self.instances:
        logger.info("No instances to display.")
        return

    instance_ids = [instance["InstanceId"] for instance in self.instances]
    paginator = self.ec2_client.get_paginator("describe_instances")
    page_iterator = paginator.paginate(InstanceIds=instance_ids)

    try:
        for page in page_iterator:
            for reservation in page["Reservations"]:
                for instance in reservation["Instances"]:
                    instance_state = instance["State"]["Name"]

                    # Apply the state filter (default is 'running')
                    if state_filter and instance_state != state_filter:
                        continue # Skip this instance if it doesn't match
the filter

                    # Create a formatted string with instance details
                    instance_info = (
                        f"• ID: {instance['InstanceId']}\n"
                        f"• Image ID: {instance['ImageId']}\n"
                        f"• Instance type: {instance['InstanceType']}\n"
                        f"• Key name: {instance['KeyName']}\n"
                        f"• VPC ID: {instance['VpcId']}\n"

```



```
        f"• Public IP: {instance.get('PublicIpAddress', 'N/A')}\n"
        f"• State: {instance_state}"
    )
    print(instance_info)

except ClientError as err:
    logger.error(
        f"Failed to display instance(s). : {' '.join(map(str, instance_ids))}"
    )
    error_code = err.response["Error"]["Code"]
    if error_code == "InvalidInstanceID.NotFound":
        logger.error(
            "One or more instance IDs do not exist. "
            "Please verify the instance IDs and try again."
        )
        raise

def terminate(self) -> None:
    """
    Terminates instances and waits for them to reach the terminated state.
    """
    if not self.instances:
        logger.info("No instances to terminate.")
        return

    instance_ids = [instance["InstanceId"] for instance in self.instances]
    try:
        self.ec2_client.terminate_instances(InstanceIds=instance_ids)
        waiter = self.ec2_client.get_waiter("instance_terminated")
        waiter.wait(InstanceIds=instance_ids)
        self.instances.clear()
        for instance_id in instance_ids:
            print(f"• Instance ID: {instance_id}\n" f"• Action: Terminated")

    except ClientError as err:
        logger.error(
            f"Failed instance termination details:\n\t{str(self.instances)}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidInstanceID.NotFound":
            logger.error(
```

```
        "One or more instance IDs do not exist. "  
        "Please verify the instance IDs and try again."  
    )  
    raise  
  
def start(self) -> Optional[Dict[str, Any]]:  
    """  
    Starts instances and waits for them to be in a running state.  
  
    :return: The response to the start request.  
    """  
    if not self.instances:  
        logger.info("No instances to start.")  
        return None  
  
    instance_ids = [instance["InstanceId"] for instance in self.instances]  
    try:  
        start_response =  
self.ec2_client.start_instances(InstanceIds=instance_ids)  
        waiter = self.ec2_client.get_waiter("instance_running")  
        waiter.wait(InstanceIds=instance_ids)  
        return start_response  
    except ClientError as err:  
        logger.error(  
            f"Failed to start instance(s): {' '.join(map(str,  
instance_ids))}"  
        )  
        error_code = err.response["Error"]["Code"]  
        if error_code == "IncorrectInstanceState":  
            logger.error(  
                "Couldn't start instance(s) because they are in an incorrect  
state. "  
                "Ensure the instances are in a stopped state before starting  
them."  
            )  
            raise  
  
def stop(self) -> Optional[Dict[str, Any]]:  
    """  
    Stops instances and waits for them to be in a stopped state.
```

```

        :return: The response to the stop request, or None if there are no
instances to stop.
        """
        if not self.instances:
            logger.info("No instances to stop.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            # Attempt to stop the instances
            stop_response =
self.ec2_client.stop_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_stopped")
            waiter.wait(InstanceIds=instance_ids)
        except ClientError as err:
            logger.error(
                f"Failed to stop instance(s): {','.join(map(str, instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't stop instance(s) because they are in an incorrect
state. "
                    "Ensure the instances are in a running state before stopping
them."
                )
            raise
        return stop_response

    def get_images(self, image_ids: List[str]) -> List[Dict[str, Any]]:
        """
        Gets information about Amazon Machine Images (AMIs) from a list of AMI
IDs.

        :param image_ids: The list of AMI IDs to look up.
        :return: A list of dictionaries representing the requested AMIs.
        """
        try:
            response = self.ec2_client.describe_images(ImageIds=image_ids)
            images = response["Images"]
        except ClientError as err:
            logger.error(f"Failed to stop AMI(s): {','.join(map(str,
image_ids))}")

```

```

        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidAMIID.NotFound":
            logger.error("One or more of the AMI IDs does not exist.")
            raise
    return images

def get_instance_types(
    self, architecture: str = "x86_64", sizes: List[str] = ["*.micro",
    "*.small"]
) -> List[Dict[str, Any]]:
    """
    Gets instance types that support the specified architecture and size.
    See https://docs.aws.amazon.com/AWSEC2/latest/APIReference/
    API\_DescribeInstanceTypes.html
    for a list of allowable parameters.

    :param architecture: The architecture supported by instance types.
    Default: 'x86_64'.
    :param sizes: The size of instance types. Default: '*.micro', '*.small',
    :return: A list of dictionaries representing instance types that support
    the specified architecture and size.
    """
    try:
        inst_types = []
        paginator = self.ec2_client.get_paginator("describe_instance_types")
        for page in paginator.paginate(
            Filters=[
                {
                    "Name": "processor-info.supported-architecture",
                    "Values": [architecture],
                },
                {"Name": "instance-type", "Values": sizes},
            ]
        ):
            inst_types += page["InstanceTypes"]
    except ClientError as err:
        logger.error(
            f"Failed to get instance types: {architecture},
            {'.'.join(map(str, sizes))}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidParameterValue":
            logger.error(

```

```

        "Parameters are invalid. "
        "Ensure architecture and size strings conform to
DescribeInstanceTypes API reference."
    )
    raise
else:
    return inst_types

```

Definieren Sie eine Klasse, die Elastic-IP-Aktionen umschließt.

```

class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
            = None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

        def __init__(self, ec2_client: Any) -> None:
            """
            Initializes the ElasticIpWrapper with an EC2 client.

            :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
            access to AWS EC2 services.

```

```
    """
    self.ec2_client = ec2_client
    self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def allocate(self) -> "ElasticIpWrapper.ElasticIp":
        """
        Allocates an Elastic IP address that can be associated with an Amazon EC2
instance. By using an Elastic IP address, you can keep the public IP
address
constant even when you restart the associated instance.

        :return: The ElasticIp object for the newly created Elastic IP address.
        :raises ClientError: If the allocation fails, such as reaching the
maximum limit of Elastic IPs.
        """
        try:
            response = self.ec2_client.allocate_address(Domain="vpc")
            elastic_ip = self.ElasticIp(
                allocation_id=response["AllocationId"],
                public_ip=response["PublicIp"]
            )
            self.elastic_ips.append(elastic_ip)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AddressLimitExceeded":
                logger.error(
                    "Max IP's reached. Release unused addresses or contact AWS
Support for an increase."
                )
            raise err
        return elastic_ip
```

```
def associate(
    self, allocation_id: str, instance_id: str
) -> Union[Dict[str, Any], None]:
    """
    Associates an Elastic IP address with an instance. When this association
    is
    created, the Elastic IP's public IP address is immediately used as the
    public
    IP address of the associated instance.

    :param allocation_id: The allocation ID of the Elastic IP.
    :param instance_id: The ID of the Amazon EC2 instance.
    :return: A response that contains the ID of the association, or None if
    no Elastic IP is found.
    :raises ClientError: If the association fails, such as when the instance
    ID is not found.
    """
    elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
    allocation_id)
    if elastic_ip is None:
        logger.info(f"No Elastic IP found with allocation ID
    {allocation_id}.")
        return None

    try:
        response = self.ec2_client.associate_address(
            AllocationId=allocation_id, InstanceId=instance_id
        )
        elastic_ip.instance_id = (
            instance_id # Track the instance associated with this Elastic
    IP.
        )
    except ClientError as err:
        if err.response["Error"]["Code"] == "InvalidInstanceID.NotFound":
            logger.error(
                f"Failed to associate Elastic IP {allocation_id} with
    {instance_id} "
                "because the specified instance ID does not exist or has not
    propagated fully. "
                "Verify the instance ID and try again, or wait a few moments
    before attempting to "
                "associate the Elastic IP address."
            )
            raise
```

```
return response

def disassociate(self, allocation_id: str) -> None:
    """
    Removes an association between an Elastic IP address and an instance.
    When the
    association is removed, the instance is assigned a new public IP address.

    :param allocation_id: The allocation ID of the Elastic IP to
    disassociate.
    :raises ClientError: If the disassociation fails, such as when the
    association ID is not found.
    """
    elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
    allocation_id)
    if elastic_ip is None or elastic_ip.instance_id is None:
        logger.info(
            f"No association found for Elastic IP with allocation ID
    {allocation_id}."
        )
        return

    try:
        # Retrieve the association ID before disassociating
        response =
    self.ec2_client.describe_addresses(AllocationIds=[allocation_id])
        association_id = response["Addresses"][0].get("AssociationId")

        if association_id:

    self.ec2_client.disassociate_address(AssociationId=association_id)
            elastic_ip.instance_id = None # Remove the instance association
        else:
            logger.info(
                f"No Association ID found for Elastic IP with allocation ID
    {allocation_id}."
            )

    except ClientError as err:
        if err.response["Error"]["Code"] == "InvalidAssociationID.NotFound":
            logger.error(
                f"Failed to disassociate Elastic IP {allocation_id} "
```



```

        "because the specified association ID for the Elastic IP
address was not found. "
        "Verify the association ID and ensure the Elastic IP is
currently associated with a "
        "resource before attempting to disassociate it."
    )
    raise

def release(self, allocation_id: str) -> None:
    """
    Releases an Elastic IP address. After the Elastic IP address is released,
    it can no longer be used.

    :param allocation_id: The allocation ID of the Elastic IP to release.
    :raises ClientError: If the release fails, such as when the Elastic IP
address is not found.
    """
    elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
    if elastic_ip is None:
        logger.info(f"No Elastic IP found with allocation ID
{allocation_id}.")
        return

    try:
        self.ec2_client.release_address(AllocationId=allocation_id)
        self.elastic_ips.remove(elastic_ip) # Remove the Elastic IP from the
list
    except ClientError as err:
        if err.response["Error"]["Code"] == "InvalidAddress.NotFound":
            logger.error(
                f"Failed to release Elastic IP address {allocation_id} "
                "because it could not be found. Verify the Elastic IP address
"
                "and ensure it is allocated to your account in the correct
region "
                "before attempting to release it."
            )
            raise

    @staticmethod
    def get_elastic_ip_by_allocation(

```

```
    elastic_ips: List["ElasticIpWrapper.ElasticIp"], allocation_id: str
) -> Optional["ElasticIpWrapper.ElasticIp"]:
    """
    Retrieves an Elastic IP object by its allocation ID from a given list of
    Elastic IPs.

    :param elastic_ips: A list of ElasticIp objects.
    :param allocation_id: The allocation ID of the Elastic IP to retrieve.
    :return: The ElasticIp object associated with the allocation ID, or None
    if not found.
    """
    return next(
        (ip for ip in elastic_ips if ip.allocation_id == allocation_id), None
    )
```

- API-Einheiten finden Sie in der Python-Referenz (Boto3) API in AWS SDK den folgenden Themen.
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)
 - [DeleteSecurityGroup](#)
 - [DescribeImages](#)
 - [DescribeInstanceTypes](#)
 - [DescribeInstances](#)
 - [DescribeKeyPairs](#)
 - [DescribeSecurityGroups](#)
 - [DisassociateAddress](#)
 - [ReleaseAddress](#)
 - [RunInstances](#)
 - [StartInstances](#)

- [StopInstances](#)
- [TerminateInstances](#)
- [UnmonitorInstances](#)

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Die EC2InstanceScenario Implementierung enthält Logik, um das Beispiel als Ganzes auszuführen.

```
#!/ Scenario that uses the AWS SDK for Rust (the SDK) with Amazon Elastic Compute
Cloud
#!/ (Amazon EC2) to do the following:
#!/
#!/ * Create a key pair that is used to secure SSH communication between your
computer and
#!/ an EC2 instance.
#!/ * Create a security group that acts as a virtual firewall for your EC2
instances to
#!/ control incoming and outgoing traffic.
#!/ * Find an Amazon Machine Image (AMI) and a compatible instance type.
#!/ * Create an instance that is created from the instance type and AMI you
select, and
#!/ is configured to use the security group and key pair created in this
example.
#!/ * Stop and restart the instance.
#!/ * Create an Elastic IP address and associate it as a consistent IP address
for your instance.
#!/ * Connect to your instance with SSH, using both its public IP address and
your Elastic IP
address.
#!/ * Clean up all of the resources created by this example.
```

```
use std::net::Ipv4Addr;

use crate::{
    ec2::{EC2Error, EC2},
    getting_started::{key_pair::KeyPairManager, util::Util},
    ssm::SSM,
};
use aws_sdk_ssm::types::Parameter;

use super::{
    elastic_ip::ElasticIpManager, instance::InstanceManager,
    security_group::SecurityGroupManager,
    util::ScenarioImage,
};

pub struct Ec2InstanceScenario {
    ec2: EC2,
    ssm: SSM,
    util: Util,
    key_pair_manager: KeyPairManager,
    security_group_manager: SecurityGroupManager,
    instance_manager: InstanceManager,
    elastic_ip_manager: ElasticIpManager,
}

impl Ec2InstanceScenario {
    pub fn new(ec2: EC2, ssm: SSM, util: Util) -> Self {
        Ec2InstanceScenario {
            ec2,
            ssm,
            util,
            key_pair_manager: Default::default(),
            security_group_manager: Default::default(),
            instance_manager: Default::default(),
            elastic_ip_manager: Default::default(),
        }
    }

    pub async fn run(&mut self) -> Result<(), EC2Error> {
        self.create_and_list_key_pairs().await?;
        self.create_security_group().await?;
        self.create_instance().await?;
        self.stop_and_start_instance().await?;
        self.associate_elastic_ip().await?;
    }
}
```

```
        self.stop_and_start_instance().await?;
        Ok(())
    }

    /// 1. Creates an RSA key pair and saves its private key data as a .pem file
    in secure
    ///    temporary storage. The private key data is deleted after the example
    completes.
    /// 2. Optionally, lists the first five key pairs for the current account.
    pub async fn create_and_list_key_pairs(&mut self) -> Result<(), EC2Error> {
        println!( "Let's create an RSA key pair that you can be use to securely
        connect to your EC2 instance.");

        let key_name = self.util.prompt_key_name()?;

        self.key_pair_manager
            .create(&self.ec2, &self.util, key_name)
            .await?;

        println!(
            "Created a key pair {} and saved the private key to {:?}.",
            self.key_pair_manager
                .key_pair()
                .key_name()
                .ok_or_else(|| EC2Error::new("No key name after creating key"))?,
            self.key_pair_manager
                .key_file_path()
                .ok_or_else(|| EC2Error::new("No key file after creating key"))?
        );

        if self.util.should_list_key_pairs()? {
            for pair in self.key_pair_manager.list(&self.ec2).await? {
                println!(
                    "Found {:?} key {} with fingerprint:\t{:?}",
                    pair.key_type(),
                    pair.key_name().unwrap_or("Unknown"),
                    pair.key_fingerprint()
                );
            }
        }

        Ok(())
    }
}
```

```
/// 1. Creates a security group for the default VPC.
/// 2. Adds an inbound rule to allow SSH. The SSH rule allows only
///    inbound traffic from the current computer's public IPv4 address.
/// 3. Displays information about the security group.
///
/// This function uses <http://checkip.amazonaws.com> to get the current
public IP
/// address of the computer that is running the example. This method works in
most
/// cases. However, depending on how your computer connects to the internet,
you
/// might have to manually add your public IP address to the security group
by using
/// the AWS Management Console.
pub async fn create_security_group(&mut self) -> Result<(), EC2Error> {
    println!("Let's create a security group to manage access to your
instance.");
    let group_name = self.util.prompt_security_group_name()?;

    self.security_group_manager
        .create(
            &self.ec2,
            &group_name,
            "Security group for example: get started with instances.",
        )
        .await?;

    println!(
        "Created security group {} in your default VPC {}.",
        self.security_group_manager.group_name(),
        self.security_group_manager
            .vpc_id()
            .unwrap_or("(unknown vpc)")
    );

    let check_ip = self.util.do_get("https://checkip.amazonaws.com").await?;
    let current_ip_address: Ipv4Addr = check_ip.trim().parse().map_err(|e| {
        EC2Error::new(format!(
            "Failed to convert response {} to IP Address: {e:?}",
            check_ip
        ))
    })?;

    println!("Your public IP address seems to be {current_ip_address}");
```

```

    if self.util.should_add_to_security_group() {
        match self
            .security_group_manager
            .authorize_ingress(&self.ec2, current_ip_address)
            .await
        {
            Ok(_) => println!("Security group rules updated"),
            Err(err) => eprintln!("Couldn't update security group rules:
{err:?}"),
        }
    }
    println!("{}", self.security_group_manager);

    Ok(())
}

/// 1. Gets a list of Amazon Linux 2 AMIs from AWS Systems Manager.
Specifying the
/// '/aws/service/ami-amazon-linux-latest' path returns only the latest
AMIs.
/// 2. Gets and displays information about the available AMIs and lets you
select one.
/// 3. Gets a list of instance types that are compatible with the selected
AMI and
/// lets you select one.
/// 4. Creates an instance with the previously created key pair and security
group,
/// and the selected AMI and instance type.
/// 5. Waits for the instance to be running and then displays its
information.
pub async fn create_instance(&mut self) -> Result<(), EC2Error> {
    let ami = self.find_image().await?;

    let instance_types = self
        .ec2
        .list_instance_types(&ami.0)
        .await
        .map_err(|e| e.add_message("Could not find instance types"))?;
    println!(
        "There are several instance types that support the {} architecture of
the image.",
        ami.0
            .architecture
            .as_ref()

```

```

        .ok_or_else(|| EC2Error::new(format!("Missing architecture in
{:?}", ami.0)))?
    );
    let instance_type = self.util.select_instance_type(instance_types)?;

    println!("Creating your instance and waiting for it to start...");
    self.instance_manager
        .create(
            &self.ec2,
            ami.0
                .image_id()
                .ok_or_else(|| EC2Error::new("Could not find image ID"))?,
            instance_type,
            self.key_pair_manager.key_pair(),
            self.security_group_manager
                .security_group()
                .map(|sg| vec![sg])
                .ok_or_else(|| EC2Error::new("Could not find security
group"))?,
        )
        .await
        .map_err(|e| e.add_message("Scenario failed to create instance"))?;

    while let Err(err) = self
        .ec2
        .wait_for_instance_ready(self.instance_manager.instance_id(), None)
        .await
    {
        println!("{err}");
        if !self.util.should_continue_waiting() {
            return Err(err);
        }
    }

    println!("Your instance is ready:\n{}", self.instance_manager);

    self.display_ssh_info();

    Ok(())
}

async fn find_image(&mut self) -> Result<ScenarioImage, EC2Error> {
    let params: Vec<Parameter> = self
        .ssm

```



```
        .list_path("/aws/service/ami-amazon-linux-latest")
        .await
        .map_err(|e| e.add_message("Could not find parameters for available
images")))?
        .into_iter()
        .filter(|param| param.name().is_some_and(|name|
name.contains("amzn2")))
        .collect();
    let amzn2_images: Vec<ScenarioImage> = self
        .ec2
        .list_images(params)
        .await
        .map_err(|e| e.add_message("Could not find images")))?
        .into_iter()
        .map(ScenarioImage::from)
        .collect();
    println!("We will now create an instance from an Amazon Linux 2 AMI");
    let ami = self.util.select_scenario_image(amzn2_images)?;
    Ok(ami)
}

// 1. Stops the instance and waits for it to stop.
// 2. Starts the instance and waits for it to start.
// 3. Displays information about the instance.
// 4. Displays an SSH connection string. When an Elastic IP address is
associated
//    with the instance, the IP address stays consistent when the instance
stops
//    and starts.
pub async fn stop_and_start_instance(&self) -> Result<(), EC2Error> {
    println!("Let's stop and start your instance to see what changes.");
    println!("Stopping your instance and waiting until it's stopped...");
    self.instance_manager.stop(&self.ec2).await?;
    println!("Your instance is stopped. Restarting...");
    self.instance_manager.start(&self.ec2).await?;
    println!("Your instance is running.");
    println!("{}", self.instance_manager);
    if self.elastic_ip_manager.public_ip() == "0.0.0.0" {
        println!("Every time your instance is restarted, its public IP
address changes.");
    } else {
        println!(
            "Because you have associated an Elastic IP with your instance,
you can connect by using a consistent IP address after the instance restarts."
```

```

        );
    }
    self.display_ssh_info();
    Ok(())
}

/// 1. Allocates an Elastic IP address and associates it with the instance.
/// 2. Displays an SSH connection string that uses the Elastic IP address.
async fn associate_elastic_ip(&mut self) -> Result<(), EC2Error> {
    self.elastic_ip_manager.allocate(&self.ec2).await?;
    println!(
        "Allocated static Elastic IP address: {}",
        self.elastic_ip_manager.public_ip()
    );

    self.elastic_ip_manager
        .associate(&self.ec2, self.instance_manager.instance_id())
        .await?;
    println!("Associated your Elastic IP with your instance.");
    println!("You can now use SSH to connect to your instance by using the
Elastic IP.");
    self.display_ssh_info();
    Ok(())
}

/// Displays an SSH connection string that can be used to connect to a
running
/// instance.
fn display_ssh_info(&self) {
    let ip_addr = if self.elastic_ip_manager.has_allocation() {
        self.elastic_ip_manager.public_ip()
    } else {
        self.instance_manager.instance_ip()
    };
    let key_file_path = self.key_pair_manager.key_file_path().unwrap();
    println!("To connect, open another command prompt and run the following
command:");
    println!("\nssh -i {} ec2-user@{ip_addr}\n", key_file_path.display());
    let _ = self.util.enter_to_continue();
}

/// 1. Disassociate and delete the previously created Elastic IP.
/// 2. Terminate the previously created instance.
/// 3. Delete the previously created security group.

```

```
/// 4. Delete the previously created key pair.
pub async fn clean_up(self) {
    println!("Let's clean everything up. This example created these
resources:");
    println!(
        "\tKey pair: {}",
        self.key_pair_manager
            .key_pair()
            .key_name()
            .unwrap_or("(unknown key pair)")
    );
    println!(
        "\tSecurity group: {}",
        self.security_group_manager.group_name()
    );
    println!(
        "\tInstance: {}",
        self.instance_manager.instance_display_name()
    );
    if self.util.should_clean_resources() {
        if let Err(err) = self.elastic_ip_manager.remove(&self.ec2).await {
            eprintln!("{err}")
        }
        if let Err(err) = self.instance_manager.delete(&self.ec2).await {
            eprintln!("{err}")
        }
        if let Err(err) = self.security_group_manager.delete(&self.ec2).await
        {
            eprintln!("{err}");
        }
        if let Err(err) = self.key_pair_manager.delete(&self.ec2,
&self.util).await {
            eprintln!("{err}");
        }
        } else {
            println!("Ok, not cleaning up any resources!");
        }
    }
}

pub async fn run(mut scenario: Ec2InstanceScenario) {
    println!
    ("-----");
    println!(
```

```

        "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started
with instances demo."
    );
    println!
("-----");

    if let Err(err) = scenario.run().await {
        eprintln!("There was an error running the scenario: {err}")
    }

    println!
("-----");

    scenario.clean_up().await;

    println!("Thanks for running!");
    println!
("-----");
}

```

Die `EC2Impl` Struktur dient als Automock-Punkt zum Testen, und ihre Funktionen umschließen die EC2 SDK Aufrufe.

```

use std::{net::Ipv4Addr, time::Duration};

use aws_sdk_ec2::{
    client::Waiters,
    error::ProvideErrorMetadata,
    operation::{
        allocate_address::AllocateAddressOutput,
        associate_address::AssociateAddressOutput,
    },
    types::{
        DomainType, Filter, Image, Instance, InstanceType, IpPermission, IpRange,
        KeyPairInfo,
        SecurityGroup, Tag,
    },
    Client as EC2Client,
};
use aws_sdk_ssm::types::Parameter;
use aws_smithy_runtime_api::client::waiters::error::WaiterError;

```

```
#[cfg(test)]
use mockall::automock;

#[cfg(not(test))]
pub use EC2Impl as EC2;

#[cfg(test)]
pub use MockEC2Impl as EC2;

#[derive(Clone)]
pub struct EC2Impl {
    pub client: EC2Client,
}

#[cfg_attr(test, automock)]
impl EC2Impl {
    pub fn new(client: EC2Client) -> Self {
        EC2Impl { client }
    }

    pub async fn create_key_pair(&self, name: String) -> Result<(KeyPairInfo,
String), EC2Error> {
        tracing::info!("Creating key pair {name}");
        let output = self.client.create_key_pair().key_name(name).send().await?;
        let info = KeyPairInfo::builder()
            .set_key_name(output.key_name)
            .set_key_fingerprint(output.key_fingerprint)
            .set_key_pair_id(output.key_pair_id)
            .build();
        let material = output
            .key_material
            .ok_or_else(|| EC2Error::new("Create Key Pair has no key
material"))?;
        Ok((info, material))
    }

    pub async fn list_key_pair(&self) -> Result<Vec<KeyPairInfo>, EC2Error> {
        let output = self.client.describe_key_pairs().send().await?;
        Ok(output.key_pairs.unwrap_or_default())
    }

    pub async fn delete_key_pair(&self, key_pair_id: &str) -> Result<(),
EC2Error> {
```

```
    let key_pair_id: String = key_pair_id.into();
    tracing::info!("Deleting key pair {key_pair_id}");
    self.client
        .delete_key_pair()
        .key_pair_id(key_pair_id)
        .send()
        .await?;
    Ok(())
}

pub async fn create_security_group(
    &self,
    name: &str,
    description: &str,
) -> Result<SecurityGroup, EC2Error> {
    tracing::info!("Creating security group {name}");
    let create_output = self
        .client
        .create_security_group()
        .group_name(name)
        .description(description)
        .send()
        .await
        .map_err(EC2Error::from)?;

    let group_id = create_output
        .group_id
        .ok_or_else(|| EC2Error::new("Missing security group id after
creation"))?;

    let group = self
        .describe_security_group(&group_id)
        .await?
        .ok_or_else(|| {
            EC2Error::new(format!("Could not find security group with id
{group_id}"))
        })?;

    tracing::info!("Created security group {name} as {group_id}");

    Ok(group)
}
```

```
/// Find a single security group, by ID. Returns Err if multiple groups are
found.
pub async fn describe_security_group(
    &self,
    group_id: &str,
) -> Result<Option<SecurityGroup>, EC2Error> {
    let group_id: String = group_id.into();
    let describe_output = self
        .client
        .describe_security_groups()
        .group_ids(&group_id)
        .send()
        .await?;

    let mut groups = describe_output.security_groups.unwrap_or_default();

    match groups.len() {
        0 => Ok(None),
        1 => Ok(Some(groups.remove(0))),
        _ => Err(EC2Error::new(format!(
            "Expected single group for {group_id}"
        ))),
    }
}

/// Add an ingress rule to a security group explicitly allowing IPv4 address
/// as {ip}/32 over TCP port 22.
pub async fn authorize_security_group_ssh_ingress(
    &self,
    group_id: &str,
    ingress_ips: Vec<Ipv4Addr>,
) -> Result<(), EC2Error> {
    tracing::info!("Authorizing ingress for security group {group_id}");
    self.client
        .authorize_security_group_ingress()
        .group_id(group_id)
        .set_ip_permissions(Some(
            ingress_ips
                .into_iter()
                .map(|ip| {
                    IpPermission::builder()
                        .ip_protocol("tcp")
                        .from_port(22)
                        .to_port(22)
                })
        ))
}
```

```

        .ip_ranges(IpRange::builder().cidr_ip(format!
("{ip}/32")).build())
        .build()
    })
    .collect(),
))
.send()
.await?;
Ok(())
}

pub async fn delete_security_group(&self, group_id: &str) -> Result<(),
EC2Error> {
    tracing::info!("Deleting security group {group_id}");
    self.client
        .delete_security_group()
        .group_id(group_id)
        .send()
        .await?;
    Ok(())
}

pub async fn list_images(&self, ids: Vec<Parameter>) -> Result<Vec<Image>,
EC2Error> {
    let image_ids = ids.into_iter().filter_map(|p| p.value).collect();
    let output = self
        .client
        .describe_images()
        .set_image_ids(Some(image_ids))
        .send()
        .await?;

    let images = output.images.unwrap_or_default();
    if images.is_empty() {
        Err(EC2Error::new("No images for selected AMIs"))
    } else {
        Ok(images)
    }
}

/// List instance types that match an image's architecture and are free tier
eligible.
pub async fn list_instance_types(&self, image: &Image) ->
Result<Vec<InstanceType>, EC2Error> {

```



```
let architecture = format!(
    "{}",
    image.architecture().ok_or_else(|| EC2Error::new(format!(
        "Image {:?} does not have a listed architecture",
        image.image_id()
    )))?
);
let free_tier_eligible_filter = Filter::builder()
    .name("free-tier-eligible")
    .values("false")
    .build();
let supported_architecture_filter = Filter::builder()
    .name("processor-info.supported-architecture")
    .values(architecture)
    .build();
let response = self
    .client
    .describe_instance_types()
    .filters(free_tier_eligible_filter)
    .filters(supported_architecture_filter)
    .send()
    .await?;

Ok(response
    .instance_types
    .unwrap_or_default()
    .into_iter()
    .filter_map(|iti| iti.instance_type)
    .collect())
}

pub async fn create_instance<'a>(
    &self,
    image_id: &'a str,
    instance_type: InstanceType,
    key_pair: &'a KeyPairInfo,
    security_groups: Vec<&'a SecurityGroup>,
) -> Result<String, EC2Error> {
    let run_instances = self
        .client
        .run_instances()
        .image_id(image_id)
        .instance_type(instance_type)
        .key_name(
```

```
        key_pair
            .key_name()
            .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?,
    )
    .set_security_group_ids(Some(
        security_groups
            .iter()
            .filter_map(|sg| sg.group_id.clone())
            .collect(),
    ))
    .min_count(1)
    .max_count(1)
    .send()
    .await?;

if run_instances.instances().is_empty() {
    return Err(EC2Error::new("Failed to create instance"));
}

let instance_id = run_instances.instances()[0].instance_id().unwrap();
let response = self
    .client
    .create_tags()
    .resources(instance_id)
    .tags(
        Tag::builder()
            .key("Name")
            .value("From SDK Examples")
            .build(),
    )
    .send()
    .await;

match response {
    Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
    Err(err) => {
        tracing::info!("Error applying tags to {instance_id}: {err:?}");
        return Err(err.into());
    }
}

tracing::info!("Instance is created.");
```

```

    Ok(instance_id.to_string())
}

/// Wait for an instance to be ready and status ok (default wait 60 seconds)
pub async fn wait_for_instance_ready(
    &self,
    instance_id: &str,
    duration: Option<Duration>,
) -> Result<(), EC2Error> {
    self.client
        .wait_until_instance_status_ok()
        .instance_ids(instance_id)
        .wait(duration.unwrap_or(Duration::from_secs(60)))
        .await
        .map_err(|err| match err {
            WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                "Exceeded max time ({}s) waiting for instance to start.",
                exceeded.max_wait().as_secs()
            )),
            _ => EC2Error::from(err),
        })?;
    Ok(())
}

pub async fn describe_instance(&self, instance_id: &str) -> Result<Instance,
EC2Error> {
    let response = self
        .client
        .describe_instances()
        .instance_ids(instance_id)
        .send()
        .await?;

    let instance = response
        .reservations()
        .first()
        .ok_or_else(|| EC2Error::new(format!("No instance reservations for
{instance_id}")))?
        .instances()
        .first()
        .ok_or_else(|| {
            EC2Error::new(format!("No instances in reservation for
{instance_id}"))
        })?;
}

```

```
        Ok(instance.clone())
    }

    pub async fn start_instance(&self, instance_id: &str) -> Result<(), EC2Error>
    {
        tracing::info!("Starting instance {instance_id}");

        self.client
            .start_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        tracing::info!("Started instance.");

        Ok(())
    }

    pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error>
    {
        tracing::info!("Stopping instance {instance_id}");

        self.client
            .stop_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        self.wait_for_instance_stopped(instance_id, None).await?;

        tracing::info!("Stopped instance.");

        Ok(())
    }

    pub async fn reboot_instance(&self, instance_id: &str) -> Result<(),
    EC2Error> {
        tracing::info!("Rebooting instance {instance_id}");

        self.client
            .reboot_instances()
            .instance_ids(instance_id)
            .send()
```

```
        .await?;

    Ok(())
}

pub async fn wait_for_instance_stopped(
    &self,
    instance_id: &str,
    duration: Option<Duration>,
) -> Result<(), EC2Error> {
    self.client
        .wait_until_instance_stopped()
        .instance_ids(instance_id)
        .wait(duration.unwrap_or(Duration::from_secs(60)))
        .await
        .map_err(|err| match err {
            WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                "Exceeded max time ({}s) waiting for instance to stop.",
                exceeded.max_wait().as_secs(),
            )),
            _ => EC2Error::from(err),
        })?;
    Ok(())
}

pub async fn delete_instance(&self, instance_id: &str) -> Result<(),
EC2Error> {
    tracing::info!("Deleting instance with id {instance_id}");
    self.stop_instance(instance_id).await?;
    self.client
        .terminate_instances()
        .instance_ids(instance_id)
        .send()
        .await?;
    self.wait_for_instance_terminated(instance_id).await?;
    tracing::info!("Terminated instance with id {instance_id}");
    Ok(())
}

async fn wait_for_instance_terminated(&self, instance_id: &str) -> Result<(),
EC2Error> {
    self.client
        .wait_until_instance_terminated()
        .instance_ids(instance_id)
```

```
        .wait(Duration::from_secs(60))
        .await
        .map_err(|err| match err {
            WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                "Exceeded max time ({}s) waiting for instance to terminate.",
                exceeded.max_wait().as_secs(),
            )),
            _ => EC2Error::from(err),
        })?;
    Ok(())
}

pub async fn allocate_ip_address(&self) -> Result<AllocateAddressOutput,
EC2Error> {
    self.client
        .allocate_address()
        .domain(DomainType::Vpc)
        .send()
        .await
        .map_err(EC2Error::from)
}

pub async fn deallocate_ip_address(&self, allocation_id: &str) -> Result<(),
EC2Error> {
    self.client
        .release_address()
        .allocation_id(allocation_id)
        .send()
        .await?;
    Ok(())
}

pub async fn associate_ip_address(
    &self,
    allocation_id: &str,
    instance_id: &str,
) -> Result<AssociateAddressOutput, EC2Error> {
    let response = self
        .client
        .associate_address()
        .allocation_id(allocation_id)
        .instance_id(instance_id)
        .send()
        .await?;
```

```
        Ok(response)
    }

    pub async fn disassociate_ip_address(&self, association_id: &str) ->
Result<(), EC2Error> {
        self.client
            .disassociate_address()
            .association_id(association_id)
            .send()
            .await?;
        Ok(())
    }
}

#[derive(Debug)]
pub struct EC2Error(String);
impl EC2Error {
    pub fn new(value: impl Into<String>) -> Self {
        EC2Error(value.into())
    }

    pub fn add_message(self, message: impl Into<String>) -> Self {
        EC2Error(format!("{}: {}", message.into(), self.0))
    }
}

impl<T: ProvideErrorMetadata> From<T> for EC2Error {
    fn from(value: T) -> Self {
        EC2Error(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 EC2Error {}
```

```
impl std::fmt::Display for EC2Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
```

Die SSM Struktur dient als Automock-Punkt für Tests, und ihre Funktionen SSM SDK umschließen Aufrufe.

```
use aws_sdk_ssm::{types::Parameter, Client};
use aws_smithy_async::future::pagination_stream::TryFlatMap;

use crate::ec2::EC2Error;

#[cfg(test)]
use mockall::automock;

#[cfg(not(test))]
pub use SSMImpl as SSM;

#[cfg(test)]
pub use MockSSMImpl as SSM;

pub struct SSMImpl {
    inner: Client,
}

#[cfg_attr(test, automock)]
impl SSMImpl {
    pub fn new(inner: Client) -> Self {
        SSMImpl { inner }
    }

    pub async fn list_path(&self, path: &str) -> Result<Vec<Parameter>, EC2Error>
    {
        let maybe_params: Vec<Result<Parameter, _>> = TryFlatMap::new(
            self.inner
                .get_parameters_by_path()
                .path(path)
                .into_paginator()
                .send(),
```



```

    )
    .flat_map(|item| item.parameters.unwrap_or_default())
    .collect()
    .await;
    // Fail on the first error
    let params = maybe_params
        .into_iter()
        .collect::<Result<Vec<Parameter>, _>>()?;
    Ok(params)
}
}

```

Das Szenario verwendet mehrere Strukturen im „Manager“-Stil, um den Zugriff auf Ressourcen zu regeln, die im gesamten Szenario erstellt und gelöscht werden.

```

use aws_sdk_ec2::operation::{
    allocate_address::AllocateAddressOutput,
    associate_address::AssociateAddressOutput,
};

use crate::ec2::{EC2Error, EC2};

/// ElasticIpManager tracks the lifecycle of a public IP address, including its
/// allocation from the global pool and association with a specific instance.
#[derive(Debug, Default)]
pub struct ElasticIpManager {
    elastic_ip: Option<AllocateAddressOutput>,
    association: Option<AssociateAddressOutput>,
}

impl ElasticIpManager {
    pub fn has_allocation(&self) -> bool {
        self.elastic_ip.is_some()
    }

    pub fn public_ip(&self) -> &str {
        if let Some(allocation) = &self.elastic_ip {
            if let Some(addr) = allocation.public_ip() {
                return addr;
            }
        }
    }
}

```

```
    "0.0.0.0"
  }

  pub async fn allocate(&mut self, ec2: &EC2) -> Result<(), EC2Error> {
    let allocation = ec2.allocate_ip_address().await?;
    self.elastic_ip = Some(allocation);
    Ok(())
  }

  pub async fn associate(&mut self, ec2: &EC2, instance_id: &str) -> Result<(),
EC2Error> {
    if let Some(allocation) = &self.elastic_ip {
      if let Some(allocation_id) = allocation.allocation_id() {
        let association = ec2.associate_ip_address(allocation_id,
instance_id).await?;
        self.association = Some(association);
        return Ok(());
      }
    }
    Err(EC2Error::new("No ip address allocation to associate"))
  }

  pub async fn remove(mut self, ec2: &EC2) -> Result<(), EC2Error> {
    if let Some(association) = &self.association {
      if let Some(association_id) = association.association_id() {
        ec2.disassociate_ip_address(association_id).await?;
      }
    }
    self.association = None;
    if let Some(allocation) = &self.elastic_ip {
      if let Some(allocation_id) = allocation.allocation_id() {
        ec2.deallocate_ip_address(allocation_id).await?;
      }
    }
    self.elastic_ip = None;
    Ok(())
  }
}

use std::fmt::Display;

use aws_sdk_ec2::types::{Instance, InstanceType, KeyPairInfo, SecurityGroup};
```

```
use crate::ec2::{EC2Error, EC2};

/// InstanceManager wraps the lifecycle of an EC2 Instance.
#[derive(Debug, Default)]
pub struct InstanceManager {
    instance: Option<Instance>,
}

impl InstanceManager {
    pub fn instance_id(&self) -> &str {
        if let Some(instance) = &self.instance {
            if let Some(id) = instance.instance_id() {
                return id;
            }
        }
        "Unknown"
    }

    pub fn instance_name(&self) -> &str {
        if let Some(instance) = &self.instance {
            if let Some(tag) = instance.tags().iter().find(|e| e.key() ==
Some("Name")) {
                if let Some(value) = tag.value() {
                    return value;
                }
            }
        }
        "Unknown"
    }

    pub fn instance_ip(&self) -> &str {
        if let Some(instance) = &self.instance {
            if let Some(public_ip_address) = instance.public_ip_address() {
                return public_ip_address;
            }
        }
        "0.0.0.0"
    }

    pub fn instance_display_name(&self) -> String {
        format!("{}", self.instance_name(), self.instance_id())
    }

    /// Create an EC2 instance with the given ID on a given type, using a
```

```
/// generated KeyPair and applying a list of security groups.
pub async fn create(
    &mut self,
    ec2: &EC2,
    image_id: &str,
    instance_type: InstanceType,
    key_pair: &KeyPairInfo,
    security_groups: Vec<&SecurityGroup>,
) -> Result<(), EC2Error> {
    let instance_id = ec2
        .create_instance(image_id, instance_type, key_pair, security_groups)
        .await?;
    let instance = ec2.describe_instance(&instance_id).await?;
    self.instance = Some(instance);
    Ok(())
}

/// Start the managed EC2 instance, if present.
pub async fn start(&self, ec2: &EC2) -> Result<(), EC2Error> {
    if self.instance.is_some() {
        ec2.start_instance(self.instance_id()).await?;
    }
    Ok(())
}

/// Stop the managed EC2 instance, if present.
pub async fn stop(&self, ec2: &EC2) -> Result<(), EC2Error> {
    if self.instance.is_some() {
        ec2.stop_instance(self.instance_id()).await?;
    }
    Ok(())
}

pub async fn reboot(&self, ec2: &EC2) -> Result<(), EC2Error> {
    if self.instance.is_some() {
        ec2.reboot_instance(self.instance_id()).await?;
        ec2.wait_for_instance_stopped(self.instance_id(), None)
            .await?;
        ec2.wait_for_instance_ready(self.instance_id(), None)
            .await?;
    }
    Ok(())
}
```

```
/// Terminate and delete the managed EC2 instance, if present.
pub async fn delete(self, ec2: &EC2) -> Result<(), EC2Error> {
    if self.instance.is_some() {
        ec2.delete_instance(self.instance_id()).await?;
    }
    Ok(())
}

impl Display for InstanceManager {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        if let Some(instance) = &self.instance {
            writeln!(f, "\tID: {}",
instance.instance_id().unwrap_or("(Unknown)"));
            writeln!(
                f,
                "\tImage ID: {}",
                instance.image_id().unwrap_or("(Unknown)")
            );
            writeln!(
                f,
                "\tInstance type: {}",
                instance
                    .instance_type()
                    .map(|it| format!("{it}"))
                    .unwrap_or("(Unknown)".to_string())
            );
            writeln!(
                f,
                "\tKey name: {}",
                instance.key_name().unwrap_or("(Unknown)")
            );
            writeln!(f, "\tVPC ID: {}",
instance.vpc_id().unwrap_or("(Unknown)"));
            writeln!(
                f,
                "\tPublic IP: {}",
                instance.public_ip_address().unwrap_or("(Unknown)")
            );
            let instance_state = instance
                .state
                .as_ref()
                .map(|is| {
                    is.name()
                })
        }
    }
}
```

```
                .map(|isn| format!("{isn}"))
                .unwrap_or("(Unknown)".to_string())
            })
            .unwrap_or("(Unknown)".to_string());
        writeln!(f, "\tState: {instance_state}");
    } else {
        writeln!(f, "\tNo loaded instance");
    }
    Ok(())
}
}

use std::{env, path::PathBuf};

use aws_sdk_ec2::types::KeyPairInfo;

use crate::ec2::{EC2Error, EC2};

use super::util::Util;

/// KeyPairManager tracks a KeyPairInfo and the path the private key has been
/// written to, if it's been created.
#[derive(Debug)]
pub struct KeyPairManager {
    key_pair: KeyPairInfo,
    key_file_path: Option<PathBuf>,
    key_file_dir: PathBuf,
}

impl KeyPairManager {
    pub fn new() -> Self {
        Self::default()
    }

    pub fn key_pair(&self) -> &KeyPairInfo {
        &self.key_pair
    }

    pub fn key_file_path(&self) -> Option<&PathBuf> {
        self.key_file_path.as_ref()
    }

    pub fn key_file_dir(&self) -> &PathBuf {
```

```
        &self.key_file_dir
    }

    /// Creates a key pair that can be used to securely connect to an EC2
    instance.
    /// The returned key pair contains private key information that cannot be
    retrieved
    /// again. The private key data is stored as a .pem file.
    ///
    /// :param key_name: The name of the key pair to create.
    pub async fn create(
        &mut self,
        ec2: &EC2,
        util: &Util,
        key_name: String,
    ) -> Result<KeyPairInfo, EC2Error> {
        let (key_pair, material) = ec2
            .create_key_pair(key_name.clone())
            .await
            .map_err(|e| e.add_message(format!("Couldn't create key
{key_name}")))?;

        let path = self.key_file_dir.join(format!("{key_name}.pem"));

        util.write_secure(&key_name, &path, material)?;

        self.key_file_path = Some(path);
        self.key_pair = key_pair.clone();

        Ok(key_pair)
    }

    pub async fn delete(self, ec2: &EC2, util: &Util) -> Result<(), EC2Error> {
        if let Some(key_pair_id) = self.key_pair.key_pair_id() {
            ec2.delete_key_pair(key_pair_id).await?;
            if let Some(key_path) = self.key_file_path() {
                if let Err(err) = util.remove(key_path) {
                    eprintln!("Failed to remove {key_path:?} ({err:?})");
                }
            }
        }
        Ok(())
    }
}
```

```
    pub async fn list(&self, ec2: &EC2) -> Result<Vec<KeyPairInfo>, EC2Error> {
        ec2.list_key_pair().await
    }
}

impl Default for KeyPairManager {
    fn default() -> Self {
        KeyPairManager {
            key_pair: KeyPairInfo::builder().build(),
            key_file_path: Default::default(),
            key_file_dir: env::temp_dir(),
        }
    }
}

use std::net::Ipv4Addr;

use aws_sdk_ec2::types::SecurityGroup;

use crate::ec2::{EC2Error, EC2};

/// SecurityGroupManager tracks the lifecycle of a SecurityGroup for an instance,
/// including adding a rule to allow SSH from a public IP address.
#[derive(Debug, Default)]
pub struct SecurityGroupManager {
    group_name: String,
    group_description: String,
    security_group: Option<SecurityGroup>,
}

impl SecurityGroupManager {
    pub async fn create(
        &mut self,
        ec2: &EC2,
        group_name: &str,
        group_description: &str,
    ) -> Result<(), EC2Error> {
        self.group_name = group_name.into();
        self.group_description = group_description.into();

        self.security_group = Some(
            ec2.create_security_group(group_name, group_description)
                .await
        )
    }
}
```



```
        .map_err(|e| e.add_message("Couldn't create security group"))?,
    );

    Ok(())
}

pub async fn authorize_ingress(&self, ec2: &EC2, ip_address: Ipv4Addr) ->
Result<(), EC2Error> {
    if let Some(sg) = &self.security_group {
        ec2.authorize_security_group_ssh_ingress(
            sg.group_id()
                .ok_or_else(|| EC2Error::new("Missing security group ID"))?,
            vec![ip_address],
        )
        .await?;
    };

    Ok(())
}

pub async fn delete(self, ec2: &EC2) -> Result<(), EC2Error> {
    if let Some(sg) = &self.security_group {
        ec2.delete_security_group(
            sg.group_id()
                .ok_or_else(|| EC2Error::new("Missing security group ID"))?,
        )
        .await?;
    };

    Ok(())
}

pub fn group_name(&self) -> &str {
    &self.group_name
}

pub fn vpc_id(&self) -> Option<&str> {
    self.security_group.as_ref().and_then(|sg| sg.vpc_id())
}

pub fn security_group(&self) -> Option<&SecurityGroup> {
    self.security_group.as_ref()
}
}
```

```
impl std::fmt::Display for SecurityGroupManager {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        match &self.security_group {
            Some(sg) => {
                writeln!(
                    f,
                    "Security group: {}",
                    sg.group_name().unwrap_or("(unknown group)")
                )?;
                writeln!(f, "\tID: {}", sg.group_id().unwrap_or("(unknown group
id)"))?;
                writeln!(f, "\tVPC: {}", sg.vpc_id().unwrap_or("(unknown group
vpc)"))?;
                if !sg.ip_permissions().is_empty() {
                    writeln!(f, "\tInbound Permissions:");
                    for permission in sg.ip_permissions() {
                        writeln!(f, "\t\t{permission:?}")?;
                    }
                }
                Ok(())
            }
            None => writeln!(f, "No security group loaded."),
        }
    }
}
```

Der Haupteinstiegspunkt für das Szenario.

```
use ec2_code_examples::{
    ec2::EC2,
    getting_started::{
        scenario::{run, Ec2InstanceScenario},
        util::UtilImpl,
    },
    ssm::SSM,
};

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();
}
```

```
let sdk_config = aws_config::load_from_env().await;
let ec2 = EC2::new(aws_sdk_ec2::Client::new(&sdk_config));
let ssm = SSM::new(aws_sdk_ssm::Client::new(&sdk_config));
let util = UtilImpl {};
let scenario = Ec2InstanceScenario::new(ec2, ssm, util);
run(scenario).await;
}
```

- API-Einheiten finden Sie in der API-Rust-Referenz in AWS SDK den folgenden Themen.
 - [AllocateAddress](#)
 - [AssociateAddress](#)
 - [AuthorizeSecurityGroupIngress](#)
 - [CreateKeyPair](#)
 - [CreateSecurityGroup](#)
 - [DeleteKeyPair](#)
 - [DeleteSecurityGroup](#)
 - [DescribeImages](#)
 - [DescribeInstanceTypes](#)
 - [DescribeInstances](#)
 - [DescribeKeyPairs](#)
 - [DescribeSecurityGroups](#)
 - [DisassociateAddress](#)
 - [ReleaseAddress](#)
 - [RunInstances](#)
 - [StartInstances](#)
 - [StopInstances](#)
 - [TerminateInstances](#)
 - [UnmonitorInstances](#)

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Aktionen für Amazon bei EC2 der Verwendung AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie einzelne EC2 Amazon-Aktionen mit ausführen können AWS SDKs. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge nennen Amazon EC2 API und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Aktionen im Kontext finden Sie unter. [Szenarien für die EC2 Verwendung von Amazon AWS SDKs](#)

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der [Amazon Elastic Compute API Cloud-Referenz](#).

Beispiele

- [Verwenden Sie AcceptVpcPeeringConnection mit einem CLI](#)
- [Verwenden Sie es AllocateAddress mit einem AWS SDK oder CLI](#)
- [Verwenden Sie AllocateHosts mit einem CLI](#)
- [Verwenden Sie AssignPrivateIpAddresses mit einem CLI](#)
- [Verwenden Sie es AssociateAddress mit einem AWS SDK oder CLI](#)
- [Verwenden Sie AssociateDhcpOptions mit einem CLI](#)
- [Verwenden Sie AssociateRouteTable mit einem CLI](#)
- [Verwenden Sie AttachInternetGateway mit einem CLI](#)
- [Verwenden Sie AttachNetworkInterface mit einem CLI](#)
- [Verwenden Sie AttachVolume mit einem CLI](#)
- [Verwenden Sie AttachVpnGateway mit einem CLI](#)
- [Verwenden Sie AuthorizeSecurityGroupEgress mit einem CLI](#)
- [Verwenden Sie es AuthorizeSecurityGroupIngress mit einem AWS SDK oder CLI](#)
- [Verwenden Sie CancelCapacityReservation mit einem CLI](#)
- [Verwenden Sie CancelImportTask mit einem CLI](#)
- [Verwenden Sie CancelSpotFleetRequests mit einem CLI](#)
- [Verwenden Sie CancelSpotInstanceRequests mit einem CLI](#)
- [Verwenden Sie ConfirmProductInstance mit einem CLI](#)
- [Verwenden Sie CopyImage mit einem CLI](#)

- [Verwenden Sie CopySnapshot mit einem CLI](#)
- [Verwenden Sie CreateCapacityReservation mit einem CLI](#)
- [Verwenden Sie CreateCustomerGateway mit einem CLI](#)
- [Verwenden Sie CreateDhcpOptions mit einem CLI](#)
- [Verwenden Sie CreateFlowLogs mit einem CLI](#)
- [Verwenden Sie CreateImage mit einem CLI](#)
- [Verwenden Sie CreateInstanceExportTask mit einem CLI](#)
- [Verwenden Sie CreateInternetGateway mit einem CLI](#)
- [Verwenden Sie es CreateKeyPair mit einem AWS SDK oder CLI](#)
- [Verwenden Sie es CreateLaunchTemplate mit einem AWS SDK oder CLI](#)
- [Verwenden Sie CreateNetworkAcl mit einem CLI](#)
- [Verwenden Sie CreateNetworkAclEntry mit einem CLI](#)
- [Verwenden Sie CreateNetworkInterface mit einem CLI](#)
- [Verwenden Sie CreatePlacementGroup mit einem CLI](#)
- [Verwenden Sie CreateRoute mit einem CLI](#)
- [Verwenden Sie es CreateRouteTable mit einem AWS SDK oder CLI](#)
- [Verwenden Sie es CreateSecurityGroup mit einem AWS SDK oder CLI](#)
- [Verwenden Sie CreateSnapshot mit einem CLI](#)
- [Verwenden Sie CreateSpotDatafeedSubscription mit einem CLI](#)
- [Verwenden Sie es CreateSubnet mit einem AWS SDK oder CLI](#)
- [Verwenden Sie es CreateTags mit einem AWS SDK oder CLI](#)
- [Verwenden Sie CreateVolume mit einem CLI](#)
- [Verwenden Sie es CreateVpc mit einem AWS SDK oder CLI](#)
- [Verwenden Sie es CreateVpcEndpoint mit einem AWS SDK oder CLI](#)
- [Verwenden Sie CreateVpnConnection mit einem CLI](#)
- [Verwenden Sie CreateVpnConnectionRoute mit einem CLI](#)
- [Verwenden Sie CreateVpnGateway mit einem CLI](#)
- [Verwenden Sie DeleteCustomerGateway mit einem CLI](#)
- [Verwenden Sie DeleteDhcpOptions mit einem CLI](#)
- [Verwenden Sie DeleteFlowLogs mit einem CLI](#)

- [Verwenden Sie DeleteInternetGateway mit einem CLI](#)
- [Verwenden Sie es DeleteKeyPair mit einem AWS SDK oder CLI](#)
- [Verwenden Sie es DeleteLaunchTemplate mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DeleteNetworkAcl mit einem CLI](#)
- [Verwenden Sie DeleteNetworkAclEntry mit einem CLI](#)
- [Verwenden Sie DeleteNetworkInterface mit einem CLI](#)
- [Verwenden Sie DeletePlacementGroup mit einem CLI](#)
- [Verwenden Sie DeleteRoute mit einem CLI](#)
- [Verwenden Sie DeleteRouteTable mit einem CLI](#)
- [Verwenden Sie es DeleteSecurityGroup mit einem AWS SDK oder CLI](#)
- [Verwenden Sie es DeleteSnapshot mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DeleteSpotDatafeedSubscription mit einem CLI](#)
- [Verwenden Sie DeleteSubnet mit einem CLI](#)
- [Verwenden Sie DeleteTags mit einem CLI](#)
- [Verwenden Sie DeleteVolume mit einem CLI](#)
- [Verwenden Sie es DeleteVpc mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DeleteVpcEndpoint mit einem AWS SDK](#)
- [Verwenden Sie DeleteVpnConnection mit einem CLI](#)
- [Verwenden Sie DeleteVpnConnectionRoute mit einem CLI](#)
- [Verwenden Sie DeleteVpnGateway mit einem CLI](#)
- [Verwenden Sie DeregisterImage mit einem CLI](#)
- [Verwenden Sie DescribeAccountAttributes mit einem CLI](#)
- [Verwenden Sie DescribeAddresses mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeAvailabilityZones mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeBundleTasks mit einem CLI](#)
- [Verwenden Sie DescribeCapacityReservations mit einem CLI](#)
- [Verwenden Sie DescribeCustomerGateways mit einem CLI](#)
- [Verwenden Sie DescribeDhcpOptions mit einem CLI](#)
- [Verwenden Sie DescribeFlowLogs mit einem CLI](#)
- [Verwenden Sie DescribeHostReservationOfferings mit einem CLI](#)

- [Verwenden Sie DescribeHosts mit einem CLI](#)
- [Verwenden Sie DescribeIamInstanceProfileAssociations mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeDnsFormat mit einem CLI](#)
- [Verwenden Sie DescribeIdentityIdFormat mit einem CLI](#)
- [Verwenden Sie DescribeImageAttribute mit einem CLI](#)
- [Verwenden Sie DescribeImages mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeImportImageTasks mit einem CLI](#)
- [Verwenden Sie DescribeImportSnapshotTasks mit einem CLI](#)
- [Verwenden Sie DescribeInstanceAttribute mit einem CLI](#)
- [Verwenden Sie DescribeInstanceStatus mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeInstanceTypes mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeInstances mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeInternetGateways mit einem CLI](#)
- [Verwenden Sie DescribeKeyPairs mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeNetworkAcls mit einem CLI](#)
- [Verwenden Sie DescribeNetworkInterfaceAttribute mit einem CLI](#)
- [Verwenden Sie DescribeNetworkInterfaces mit einem CLI](#)
- [Verwenden Sie DescribePlacementGroups mit einem CLI](#)
- [Verwenden Sie DescribePrefixLists mit einem CLI](#)
- [Verwenden Sie DescribeRegions mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeRouteTables mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeScheduledInstanceAvailability mit einem CLI](#)
- [Verwenden Sie DescribeScheduledInstances mit einem CLI](#)
- [Verwenden Sie DescribeSecurityGroups mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeSnapshotAttribute mit einem CLI](#)
- [Verwenden Sie DescribeSnapshots mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeSpotDatafeedSubscription mit einem CLI](#)
- [Verwenden Sie DescribeSpotFleetInstances mit einem CLI](#)
- [Verwenden Sie DescribeSpotFleetRequestHistory mit einem CLI](#)
- [Verwenden Sie DescribeSpotFleetRequests mit einem CLI](#)

- [Verwenden Sie DescribeSpotInstanceRequests mit einem CLI](#)
- [Verwenden Sie DescribeSpotPriceHistory mit einem CLI](#)
- [Verwenden Sie DescribeSubnets mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeTags mit einem CLI](#)
- [Verwenden Sie DescribeVolumeAttribute mit einem CLI](#)
- [Verwenden Sie DescribeVolumeStatus mit einem CLI](#)
- [Verwenden Sie DescribeVolumes mit einem CLI](#)
- [Verwenden Sie DescribeVpcAttribute mit einem CLI](#)
- [Verwenden Sie DescribeVpcClassicLink mit einem CLI](#)
- [Verwenden Sie DescribeVpcClassicLinkDnsSupport mit einem CLI](#)
- [Verwenden Sie DescribeVpcEndpointServices mit einem CLI](#)
- [Verwenden Sie DescribeVpcEndpoints mit einem CLI](#)
- [Verwenden Sie DescribeVpcs mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DescribeVpnConnections mit einem CLI](#)
- [Verwenden Sie DescribeVpnGateways mit einem CLI](#)
- [Verwenden Sie DetachInternetGateway mit einem CLI](#)
- [Verwenden Sie DetachNetworkInterface mit einem CLI](#)
- [Verwenden Sie DetachVolume mit einem CLI](#)
- [Verwenden Sie DetachVpnGateway mit einem CLI](#)
- [Verwenden Sie DisableVgwRoutePropagation mit einem CLI](#)
- [Verwenden Sie DisableVpcClassicLink mit einem CLI](#)
- [Verwenden Sie DisableVpcClassicLinkDnsSupport mit einem CLI](#)
- [Verwenden Sie DisassociateAddress mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DisassociateRouteTable mit einem CLI](#)
- [Verwenden Sie EnableVgwRoutePropagation mit einem CLI](#)
- [Verwenden Sie EnableVolumelo mit einem CLI](#)
- [Verwenden Sie EnableVpcClassicLink mit einem CLI](#)
- [Verwenden Sie EnableVpcClassicLinkDnsSupport mit einem CLI](#)
- [Verwenden Sie GetConsoleOutput mit einem CLI](#)
- [Verwenden Sie GetHostReservationPurchasePreview mit einem CLI](#)

- [Verwenden Sie GetPasswordData mit einem AWS SDK oder CLI](#)
- [Verwenden Sie ImportImage mit einem CLI](#)
- [Verwenden Sie ImportKeyPair mit einem CLI](#)
- [Verwenden Sie ImportSnapshot mit einem CLI](#)
- [Verwenden Sie ModifyCapacityReservation mit einem CLI](#)
- [Verwenden Sie ModifyHosts mit einem CLI](#)
- [Verwenden Sie ModifyIdFormat mit einem CLI](#)
- [Verwenden Sie ModifyImageAttribute mit einem CLI](#)
- [Verwenden Sie ModifyInstanceAttribute mit einem CLI](#)
- [Verwenden Sie ModifyInstanceCreditSpecification mit einem CLI](#)
- [Verwenden Sie ModifyNetworkInterfaceAttribute mit einem CLI](#)
- [Verwenden Sie ModifyReservedInstances mit einem CLI](#)
- [Verwenden Sie ModifySnapshotAttribute mit einem CLI](#)
- [Verwenden Sie ModifySpotFleetRequest mit einem CLI](#)
- [Verwenden Sie ModifySubnetAttribute mit einem CLI](#)
- [Verwenden Sie ModifyVolumeAttribute mit einem CLI](#)
- [Verwenden Sie ModifyVpcAttribute mit einem CLI](#)
- [Verwenden Sie MonitorInstances mit einem AWS SDK oder CLI](#)
- [Verwenden Sie MoveAddressToVpc mit einem CLI](#)
- [Verwenden Sie PurchaseHostReservation mit einem CLI](#)
- [Verwenden Sie PurchaseScheduledInstances mit einem CLI](#)
- [Verwenden Sie RebootInstances mit einem AWS SDK oder CLI](#)
- [Verwenden Sie RegisterImage mit einem CLI](#)
- [Verwenden Sie RejectVpcPeeringConnection mit einem CLI](#)
- [Verwenden Sie ReleaseAddress mit einem AWS SDK oder CLI](#)
- [Verwenden Sie ReleaseHosts mit einem CLI](#)
- [Verwenden Sie ReplacelamInstanceProfileAssociation mit einem AWS SDK oder CLI](#)
- [Verwenden Sie ReplaceNetworkAclAssociation mit einem CLI](#)
- [Verwenden Sie ReplaceNetworkAclEntry mit einem CLI](#)
- [Verwenden Sie ReplaceRoute mit einem CLI](#)

- [Verwenden Sie ReplaceRouteTableAssociation mit einem CLI](#)
- [Verwenden Sie ReportInstanceStatus mit einem CLI](#)
- [Verwenden Sie RequestSpotFleet mit einem CLI](#)
- [Verwenden Sie RequestSpotInstances mit einem CLI](#)
- [Verwenden Sie ResetImageAttribute mit einem CLI](#)
- [Verwenden Sie ResetInstanceAttribute mit einem CLI](#)
- [Verwenden Sie ResetNetworkInterfaceAttribute mit einem CLI](#)
- [Verwenden Sie ResetSnapshotAttribute mit einem CLI](#)
- [Verwenden Sie RevokeSecurityGroupEgress mit einem CLI](#)
- [Verwenden Sie RevokeSecurityGroupIngress mit einem CLI](#)
- [Verwenden Sie RunInstances mit einem AWS SDK oder CLI](#)
- [Verwenden Sie RunScheduledInstances mit einem CLI](#)
- [Verwenden Sie StartInstances mit einem AWS SDK oder CLI](#)
- [Verwenden Sie StopInstances mit einem AWS SDK oder CLI](#)
- [Verwenden Sie TerminateInstances mit einem AWS SDK oder CLI](#)
- [Verwenden Sie UnassignPrivateIpAddresses mit einem CLI](#)
- [Verwenden Sie UnmonitorInstances mit einem AWS SDK oder CLI](#)
- [Verwenden Sie UpdateSecurityGroupRuleDescriptionsIngress mit einem CLI](#)

Verwenden Sie **AcceptVpcPeeringConnection** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `AcceptVpcPeeringConnection`.

CLI

AWS CLI

Um eine VPC Peering-Verbindung zu akzeptieren

In diesem Beispiel wird die angegebene VPC Peering-Verbindungsanforderung akzeptiert.

Befehl:

```
aws ec2 accept-vpc-peering-connection --vpc-peering-connection-id pcx-1a2b3c4d
```

Ausgabe:

```
{
  "VpcPeeringConnection": {
    "Status": {
      "Message": "Provisioning",
      "Code": "provisioning"
    },
    "Tags": [],
    "AccepterVpcInfo": {
      "OwnerId": "444455556666",
      "VpcId": "vpc-44455566",
      "CidrBlock": "10.0.1.0/28"
    },
    "VpcPeeringConnectionId": "pcx-1a2b3c4d",
    "RequesterVpcInfo": {
      "OwnerId": "444455556666",
      "VpcId": "vpc-111abc45",
      "CidrBlock": "10.0.0.0/28"
    }
  }
}
```

- API-Einheiten finden Sie [AcceptVpcPeeringConnection](#) in der AWS CLI Befehlsreferenz.

PowerShell**Tools für PowerShell**

Beispiel 1: Dieses Beispiel genehmigt die angeforderte VpcPeeringConnectionId Datei pcx-1dfad234b56ff78be

```
Approve-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-1dfad234b56ff78be
```

Ausgabe:

```
AccepterVpcInfo      : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
ExpirationTime       : 1/1/0001 12:00:00 AM
RequesterVpcInfo     : Amazon.EC2.Model.VpcPeeringConnectionVpcInfo
Status               : Amazon.EC2.Model.VpcPeeringConnectionStateReason
Tags                 : {}
```

```
VpcPeeringConnectionId : pcx-1dfad234b56ff78be
```

- API-Einheiten finden [AcceptVpcPeeringConnection AWS Tools for PowerShell](#) Sie unter Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie es **AllocateAddress** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `AllocateAddress`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Allocates an Elastic IP address that can be associated with an Amazon EC2
/// instance. By using an Elastic IP address, you can keep the public IP
address
// constant even when you restart the associated instance.
/// </summary>
/// <returns>The response object for the allocated address.</returns>
public async Task<AllocateAddressResponse> AllocateAddress()
{
    var request = new AllocateAddressRequest();

    try
```

```

    {
        var response = await _amazonEC2.AllocateAddressAsync(request);
        Console.WriteLine($"Allocated IP: {response.PublicIp} with allocation
ID {response.AllocationId}.");
        return response;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "AddressLimitExceeded")
        {
            // For more information on Elastic IP address quotas, see:
            // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-
ip-addresses-eip.html#using-instance-addressing-limit
            _logger.LogError($"Unable to allocate Elastic IP, address limit
exceeded. {ec2Exception.Message}");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError($"An error occurred while allocating Elastic IP.:
{ex.Message}");
        throw;
    }
}

```

- API-Einheiten finden Sie [AllocateAddress](#) unter AWS SDK for .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

#####
# function ec2_allocate_address

```

```
#
# This function allocates an Elastic IP address for use with Amazon Elastic
# Compute Cloud (Amazon EC2) instances in a specific AWS Region.
#
# Parameters:
#     -d domain - The domain for the Elastic IP address (either 'vpc' or
#     'standard').
#
# Returns:
#     The allocated Elastic IP address, or an error message if the operation
#     fails.
# And:
#     0 - If successful.
#     1 - If it fails.
#
#####
function ec2_allocate_address() {
    local domain response

    # Function to display usage information
    function usage() {
        echo "function ec2_allocate_address"
        echo "Allocates an Elastic IP address for use with Amazon Elastic Compute
        Cloud (Amazon EC2) instances in a specific AWS Region."
        echo " -d domain - The domain for the Elastic IP address (either 'vpc' or
        'standard')."
        echo ""
    }

    # Parse the command-line arguments
    while getopts "d:h" option; do
        case "${option}" in
            d) domain="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
```

```

export OPTIND=1

# Validate the input parameters
if [[ -z "$domain" ]]; then
    errecho "ERROR: You must provide a domain with the -d parameter (either 'vpc'
or 'standard')."
    return 1
fi

if [[ "$domain" != "vpc" && "$domain" != "standard" ]]; then
    errecho "ERROR: Invalid domain value. Must be either 'vpc' or 'standard'."
    return 1
fi

# Allocate the Elastic IP address
response=$(aws ec2 allocate-address \
    --domain "$domain" \
    --query "[PublicIp,AllocationId]" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports allocate-address operation failed."
    errecho "$response"
    return 1
}

echo "$response"
return 0
}

```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()

```


```
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then
        errecho " The service returned an error."
    elif [ "$err_code" == 255 ]; then
        errecho " 255 is a catch-all error."
    fi

    return 0
}
```

- API-Einheiten finden Sie [AllocateAddress](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#!/ Allocate an Elastic IP address and associate it with an Amazon Elastic
Compute Cloud
#!/ (Amazon EC2) instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param[out] publicIPAddress: String to return the public IP address.
  \param[out] allocationID: String to return the allocation ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::EC2::allocateAndAssociateAddress(const Aws::String &instanceId,
                                             Aws::String &publicIPAddress,
                                             Aws::String &allocationID,
                                             const
                                             Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::AllocateAddressRequest request;
    request.SetDomain(Aws::EC2::Model::DomainType::vpc);

    const Aws::EC2::Model::AllocateAddressOutcome outcome =
        ec2Client.AllocateAddress(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to allocate Elastic IP address:" <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }
    const Aws::EC2::Model::AllocateAddressResponse &response =
outcome.GetResult();
    allocationID = response.GetAllocationId();
    publicIPAddress = response.GetPublicIp();
}
```

```
    return true;
}
```

- API-Einheiten finden Sie [AllocateAddress](#) unter AWS SDK for C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So weisen Sie eine Elastic-IP-Adresse aus dem Adress-Pool von Amazon zu

Im folgenden `allocate-address`-Beispiel wird eine Elastic-IP-Adresse zugewiesen. Amazon EC2 wählt die Adresse aus dem Adresspool von Amazon aus.

```
aws ec2 allocate-address
```

Ausgabe:

```
{
  "PublicIp": "70.224.234.241",
  "AllocationId": "eipalloc-01435ba59eEXAMPLE",
  "PublicIpv4Pool": "amazon",
  "NetworkBorderGroup": "us-west-2",
  "Domain": "vpc"
}
```

Weitere Informationen finden Sie unter [Elastic IP-Adressen](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 2: So weisen Sie eine Elastic-IP-Adresse zu und verknüpfen sie mit einer Netzwerkrenzgruppe

Im folgenden `allocate-address`-Beispiel wird eine Elastic-IP-Adresse zugewiesen und sie der angegebenen Netzwerkrenzgruppe zugeordnet.

```
aws ec2 allocate-address \
  --network-border-group us-west-2-lax-1
```

Ausgabe:

```
{
  "PublicIp": "70.224.234.241",
  "AllocationId": "eipalloc-e03dd489ceEXAMPLE",
  "PublicIpv4Pool": "amazon",
  "NetworkBorderGroup": "us-west-2-lax-1",
  "Domain": "vpc"
}
```

Weitere Informationen finden Sie unter [Elastic IP-Adressen](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 3: So weisen Sie eine Elastic-IP-Adresse aus einem Adress-Pool zu, der Ihnen gehört

Im folgenden `allocate-address`-Beispiel wird eine Elastic-IP-Adresse aus einem Adress-Pool zugewiesen, den Sie in Ihr Amazon-Web-Services-Konto eingebunden haben. Amazon EC2 wählt die Adresse aus dem Adresspool aus.

```
aws ec2 allocate-address \
  --public-ipv4-pool ipv4pool-ec2-1234567890abcdef0
```

Ausgabe:


```
{
  "AllocationId": "eipalloc-02463d08ceEXAMPLE",
  "NetworkBorderGroup": "us-west-2",
  "CustomerOwnedIp": "18.218.95.81",
  "CustomerOwnedIpv4Pool": "ipv4pool-ec2-1234567890abcdef0",
  "Domain": "vpc"
  "NetworkBorderGroup": "us-west-2",
}
```

Weitere Informationen finden Sie unter [Elastic IP-Adressen](#) im EC2Amazon-Benutzerhandbuch.

- API-Einheiten finden Sie [AllocateAddress](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Allocates an Elastic IP address asynchronously in the VPC domain.
 *
 * @return a {@link CompletableFuture} containing the allocation ID of the
 * allocated Elastic IP address
 */
public CompletableFuture<String> allocateAddressAsync() {
    AllocateAddressRequest allocateRequest = AllocateAddressRequest.builder()
        .domain(DomainType.VPC)
        .build();

    CompletableFuture<AllocateAddressResponse> responseFuture =
getAsyncClient().allocateAddress(allocateRequest);
    return
responseFuture.thenApply(AllocateAddressResponse::allocationId).whenComplete((result,
ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to allocate address", ex);
        }
    });
}
```

- API-Einheiten finden Sie [AllocateAddress](#) unter AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import { AllocateAddressCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Allocates an Elastic IP address to your AWS account.
 */
export const main = async () => {
  const client = new EC2Client({});
  const command = new AllocateAddressCommand({});

  try {
    const { AllocationId, PublicIp } = await client.send(command);
    console.log("A new IP address has been allocated to your account:");
    console.log(`ID: ${AllocationId} Public IP: ${PublicIp}`);
    console.log(
      "You can view your IP addresses in the AWS Management Console for Amazon EC2. Look under Network & Security > Elastic IPs",
    );
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide these values?`);
    } else {
      throw caught;
    }
  }
};

import { fileURLToPath } from "node:url";
// Call function if run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}
```

- API-Einheiten finden Sie [AllocateAddress](#) unter AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
suspend fun getAllocateAddress(instanceIdVal: String?): String? {
    val allocateRequest =
        AllocateAddressRequest {
            domain = DomainType.Vpc
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val allocateResponse = ec2.allocateAddress(allocateRequest)
        val allocationIdVal = allocateResponse.allocationId

        val request =
            AssociateAddressRequest {
                instanceId = instanceIdVal
                allocationId = allocationIdVal
            }

        val associateResponse = ec2.associateAddress(request)
        return associateResponse.associationId
    }
}
```

- API-Einheiten finden Sie [AllocateAddress](#) in der AWS SDK API Kotlin-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Elastic IP-Adresse zugewiesen, die mit einer Instance in einer verwendet werden soll. VPC

```
New-EC2Address -Domain Vpc
```

Ausgabe:

AllocationId	Domain	PublicIp
-----	-----	-----
eipalloc-12345678	vpc	198.51.100.2

Beispiel 2: In diesem Beispiel wird eine Elastic IP-Adresse zur Verwendung mit einer Instance in -Classic zugewiesen. EC2

```
New-EC2Address
```

Ausgabe:

AllocationId	Domain	PublicIp
-----	-----	-----
	standard	203.0.113.17

- API-Einheiten finden Sie unter [AllocateAddress](#) Cmdlet-Referenz. AWS Tools for PowerShell

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
    actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
            = None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

        def __init__(self, ec2_client: Any) -> None:
            """
            Initializes the ElasticIpWrapper with an EC2 client.

            :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
            level
                access to AWS EC2 services.
            """
            self.ec2_client = ec2_client
            self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

        @classmethod
        def from_client(cls) -> "ElasticIpWrapper":
            """
            Creates an ElasticIpWrapper instance with a default EC2 client.

            :return: An instance of ElasticIpWrapper initialized with the default EC2
            client.
            """
            ec2_client = boto3.client("ec2")
            return cls(ec2_client)
```



```
def allocate(self) -> "ElasticIpWrapper.ElasticIp":
    """
    Allocates an Elastic IP address that can be associated with an Amazon EC2
    instance. By using an Elastic IP address, you can keep the public IP
    address
    constant even when you restart the associated instance.

    :return: The ElasticIp object for the newly created Elastic IP address.
    :raises ClientError: If the allocation fails, such as reaching the
    maximum limit of Elastic IPs.
    """
    try:
        response = self.ec2_client.allocate_address(Domain="vpc")
        elastic_ip = self.ElasticIp(
            allocation_id=response["AllocationId"],
            public_ip=response["PublicIp"]
        )
        self.elastic_ips.append(elastic_ip)
    except ClientError as err:
        if err.response["Error"]["Code"] == "AddressLimitExceeded":
            logger.error(
                "Max IP's reached. Release unused addresses or contact AWS
                Support for an increase."
            )
            raise err
    return elastic_ip
```

- API-Einheiten finden Sie unter [AllocateAddress](#) Python (Boto3) API -Referenz. AWS SDK

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
# Creates an Elastic IP address in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @return [String] The allocation ID corresponding to the Elastic IP address.
# @example
#   puts allocate_elastic_ip_address(Aws::EC2::Client.new(region: 'us-west-2'))
def allocate_elastic_ip_address(ec2_client)
  response = ec2_client.allocate_address(domain: 'vpc')
  response.allocation_id
rescue StandardError => e
  puts "Error allocating Elastic IP address: #{e.message}"
  'Error'
end
```

- API-Einheiten finden Sie [AllocateAddress](#) unter AWS SDK for Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
pub async fn allocate_ip_address(&self) -> Result<AllocateAddressOutput,
EC2Error> {
  self.client
    .allocate_address()
    .domain(DomainType::Vpc)
    .send()
    .await
    .map_err(EC2Error::from)
}
```

- API-Einheiten finden Sie [AllocateAddress](#) in der AWS SDK API Rust-Referenz.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
TRY.  
    oo_result = lo_ec2->allocateaddress( iv_domain = 'vpc' ).    " oo_result  
is returned for testing purposes. "  
    MESSAGE 'Allocated an Elastic IP address.' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
>av_err_msg }|.  
    MESSAGE lv_error TYPE 'E'.  
ENDTRY.
```

- API Einzelheiten finden Sie [AllocateAddress](#) unter AWS SDKSAPABAPAPI als Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **AllocateHosts** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `AllocateHosts`.

CLI

AWS CLI

Beispiel 1: Um einen Dedicated Host zuzuweisen

Im folgenden `allocate-hosts` Beispiel wird ein einzelner Dedicated Host in der `eu-west-1a` Availability Zone zugewiesen, auf dem Sie Instances starten können. `m5.large` Standardmäßig akzeptiert der Dedicated Host nur Starts von Ziel-Instances und unterstützt keine Host-Wiederherstellung.

```
aws ec2 allocate-hosts \  
  --instance-type m5.large \  
  --availability-zone eu-west-1a \  
  --quantity 1
```

Ausgabe:

```
{  
  "HostIds": [  
    "h-07879acf49EXAMPLE"  
  ]  
}
```

Beispiel 2: So weisen Sie einen Dedicated Host mit aktivierter Auto-Platzierung und Host-Wiederherstellung zu

Im folgenden `allocate-hosts` Beispiel wird ein einzelner Dedicated Host in der `eu-west-1a` Availability Zone zugewiesen, wobei Auto-Platzierung und Host-Wiederherstellung aktiviert sind.

```
aws ec2 allocate-hosts \  
  --instance-type m5.large \  
  --availability-zone eu-west-1a \  
  --auto-placement on \  
  --host-recovery on \  
  --quantity 1
```

Ausgabe:

```
{  
  "HostIds": [  
    "h-07879acf49EXAMPLE"  
  ]  
}
```

Beispiel 3: Um einen Dedicated Host mit Tags zuzuweisen

Das folgende `allocate-hosts` Beispiel weist einen einzelnen Dedicated Host zu und wendet ein Tag mit einem Schlüssel mit dem Namen `purpose` und dem Wert `production` an.

```
aws ec2 allocate-hosts \  
  --instance-type m5.large \  
  --availability-zone eu-west-1a \  
  --quantity 1 \  
  --tag-specifications 'ResourceType=dedicated-  
host,Tags={Key=purpose,Value=production}'
```

Ausgabe:

```
{  
  "HostIds": [  
    "h-07879acf49EXAMPLE"  
  ]  
}
```

Weitere Informationen finden Sie unter [Allocating Dedicated Hosts](#) im Amazon Elastic Compute Cloud-Benutzerhandbuch für Linux-Instances.

- API-Einheiten finden Sie [AllocateHosts](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird Ihrem Konto ein Dedicated Host für den angegebenen Instance-Typ und die angegebene Verfügbarkeitszone zugewiesen

```
New-EC2Host -AutoPlacement on -AvailabilityZone eu-west-1b -InstanceType  
m4.xlarge -Quantity 1
```

Ausgabe:

```
h-01e23f4cd567890f3
```

- API-Einheiten finden Sie unter [AllocateHosts AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie `AssignPrivateIpAddresses` mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `AssignPrivateIpAddresses`.

CLI

AWS CLI

Um einer bestimmten sekundären privaten IP-Adresse eine Netzwerkschnittstelle zuzuweisen

In diesem Beispiel wird die angegebene sekundäre private IP-Adresse der angegebenen Netzwerkschnittstelle zugewiesen. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 assign-private-ip-addresses --network-interface-id eni-e5aa89a3 --  
private-ip-addresses 10.0.0.82
```

Um einer Netzwerkschnittstelle sekundäre private IP-Adressen zuzuweisen, die Amazon EC2 auswählt

In diesem Beispiel werden der angegebenen Netzwerkschnittstelle zwei sekundäre private IP-Adressen zugewiesen. Amazon weist diese IP-Adressen EC2 automatisch aus den verfügbaren IP-Adressen im CIDR Blockbereich des Subnetzes zu, mit dem die Netzwerkschnittstelle verknüpft ist. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 assign-private-ip-addresses --network-interface-id eni-e5aa89a3 --  
secondary-private-ip-address-count 2
```

- API-Einheiten finden Sie [AssignPrivateIpAddresses](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene sekundäre private IP-Adresse der angegebenen Netzwerkschnittstelle zugewiesen.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress  
10.0.0.82
```

Beispiel 2: In diesem Beispiel werden zwei sekundäre private IP-Adressen erstellt und der angegebenen Netzwerkschnittstelle zugewiesen.

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -  
SecondaryPrivateIpAddressCount 2
```

- API-Einheiten finden Sie unter [AssignPrivateIpAddresses AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie es **AssociateAddress** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `AssociateAddress`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>  
/// Associates an Elastic IP address with an instance. When this association  
is
```

```
    /// created, the Elastic IP's public IP address is immediately used as the
public
    /// IP address of the associated instance.
    /// </summary>
    /// <param name="allocationId">The allocation Id of an Elastic IP address.</
param>
    /// <param name="instanceId">The instance Id of the EC2 instance to
    /// associate the address with.</param>
    /// <returns>The association Id that represents
    /// the association of the Elastic IP address with an instance.</returns>
public async Task<string> AssociateAddress(string allocationId, string
instanceId)
{
    try
    {
        var request = new AssociateAddressRequest
        {
            AllocationId = allocationId,
            InstanceId = instanceId
        };

        var response = await _amazonEC2.AssociateAddressAsync(request);
        return response.AssociationId;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidInstanceId")
        {
            _logger.LogError(
                $"InstanceId is invalid, unable to associate address.
{ec2Exception.Message}");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(
            $"An error occurred while associating the Elastic IP.:
{ex.Message}");
        throw;
    }
}
```


- API-Einheiten finden Sie [AssociateAddress](#) unter AWS SDK for .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#####
# function ec2_associate_address
#
# This function associates an Elastic IP address with an Amazon Elastic Compute
# Cloud (Amazon EC2) instance.
#
# Parameters:
#   -a allocation_id - The allocation ID of the Elastic IP address to
#   associate.
#   -i instance_id - The ID of the EC2 instance to associate the Elastic IP
#   address with.
#
# Returns:
#   0 - If successful.
#   1 - If it fails.
#
#####
function ec2_associate_address() {
    local allocation_id instance_id response

    # Function to display usage information
    function usage() {
        echo "function ec2_associate_address"
        echo "Associates an Elastic IP address with an Amazon Elastic Compute Cloud
        (Amazon EC2) instance."
        echo "  -a allocation_id - The allocation ID of the Elastic IP address to
        associate."
    }
}
```

```
    echo " -i instance_id - The ID of the EC2 instance to associate the Elastic
IP address with."
    echo ""
}

# Parse the command-line arguments
while getopts "a:i:h" option; do
    case "${option}" in
        a) allocation_id="${OPTARG}" ;;
        i) instance_id="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

# Validate the input parameters
if [[ -z "$allocation_id" ]]; then
    errecho "ERROR: You must provide an allocation ID with the -a parameter."
    return 1
fi

if [[ -z "$instance_id" ]]; then
    errecho "ERROR: You must provide an instance ID with the -i parameter."
    return 1
fi

# Associate the Elastic IP address
response=$(aws ec2 associate-address \
    --allocation-id "$allocation_id" \
    --instance-id "$instance_id" \
    --query "AssociationId" \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports associate-address operation failed."
    errecho "$response"
    return 1
}
```

```

}

echo "$response"
return 0
}

```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    }
}

```

```

elif [ "$err_code" == 254 ]; then
    errecho " The service returned an error."
elif [ "$err_code" == 255 ]; then
    errecho " 255 is a catch-all error."
fi

return 0
}

```

- API-Einheiten finden Sie [AssociateAddress](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

Aws::EC2::EC2Client ec2Client(clientConfiguration);

//! Associate an Elastic IP address with an EC2 instance.
/*!
 \param instanceId: An EC2 instance ID.
 \param allocationId: An Elastic IP allocation ID.
 \param[out] associationID: String to receive the association ID.
 \param clientConfiguration: AWS client configuration.
 \return bool: True if the address was associated with the instance; otherwise,
 false.
 */
bool AwsDoc::EC2::associateAddress(const Aws::String &instanceId, const
                                   Aws::String &allocationId,
                                   Aws::String &associationID,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::AssociateAddressRequest request;
    request.SetInstanceId(instanceId);

```

```
request.SetAllocationId(allocationId);

Aws::EC2::Model::AssociateAddressOutcome outcome =
ec2Client.AssociateAddress(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to associate address " << allocationId <<
        " with instance " << instanceId << ": " <<
        outcome.GetError().GetMessage() << std::endl;
} else {
    std::cout << "Successfully associated address " << allocationId <<
        " with instance " << instanceId << std::endl;
    associationID = outcome.GetResult().GetAssociationId();
}

return outcome.IsSuccess();
}
```

- API-Einheiten finden Sie [AssociateAddress](#) unter AWS SDK for C++ API-Referenz.

CLI

AWS CLI

Um Elastic IP-Adressen in EC2 -Classic zuzuordnen

In diesem Beispiel wird eine Elastic IP-Adresse einer Instanz in EC2 -Classic zugeordnet. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 associate-address --instance-id i-07ffe74c7330ebf53 --public-  
ip 198.51.100.0
```

Um eine Elastic IP-Adresse in EC2 - zuzuordnen VPC

In diesem Beispiel wird eine Elastic IP-Adresse einer Instanz in a zugeordnet VPC.

Befehl:

```
aws ec2 associate-address --instance-id i-0b263919b6498b123 --allocation-id eipalloc-64d5890a
```

Ausgabe:

```
{  
  "AssociationId": "eipassoc-2bebb745"  
}
```

In diesem Beispiel wird eine Elastic-IP-Adresse mit einer Netzwerkschnittstelle verknüpft.

Befehl:

```
aws ec2 associate-address --allocation-id eipalloc-64d5890a --network-interface-id eni-1a2b3c4d
```

In diesem Beispiel wird eine Elastic IP mit einer privaten IP-Adresse verknüpft, die mit einer Netzwerkschnittstelle verbunden ist.

Befehl:

```
aws ec2 associate-address --allocation-id eipalloc-64d5890a --network-interface-id eni-1a2b3c4d --private-ip-address 10.0.0.85
```

- API-Einheiten finden Sie [AssociateAddress](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
```

```
* Associates an Elastic IP address with an EC2 instance asynchronously.
*
* @param instanceId    the ID of the EC2 instance to associate the Elastic
IP address with
* @param allocationId  the allocation ID of the Elastic IP address to
associate
* @return a {@link CompletableFuture} that completes with the association ID
when the operation is successful,
*         or throws a {@link RuntimeException} if the operation fails
*/
public CompletableFuture<String> associateAddressAsync(String instanceId,
String allocationId) {
    AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
        .instanceId(instanceId)
        .allocationId(allocationId)
        .build();

    CompletableFuture<AssociateAddressResponse> responseFuture =
getAsyncClient().associateAddress(associateRequest);
    return responseFuture.thenApply(response -> {
        if (response.associationId() != null) {
            return response.associationId();
        } else {
            throw new RuntimeException("Association ID is null after
associating address.");
        }
    }).whenComplete((result, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to associate address", ex);
        }
    });
}
```

- APIEinzelheiten finden Sie [AssociateAddress](#) unter AWS SDK for Java 2.x APIReferenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import { AssociateAddressCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Associates an Elastic IP address, or carrier IP address (for instances that
 * are in subnets in Wavelength Zones)
 * with an instance or a network interface.
 * @param {{ instanceId: string, allocationId: string }} options
 */
export const main = async ({ instanceId, allocationId }) => {
  const client = new EC2Client({});
  const command = new AssociateAddressCommand({
    // You need to allocate an Elastic IP address before associating it with an
    // instance.
    // You can do that with the AllocateAddressCommand.
    AllocationId: allocationId,
    // You need to create an EC2 instance before an IP address can be associated
    // with it.
    // You can do that with the RunInstancesCommand.
    InstanceId: instanceId,
  });

  try {
    const { AssociationId } = await client.send(command);
    console.log(
      `Address with allocation ID ${allocationId} is now associated with instance
      ${instanceId}.`,
      `The association ID is ${AssociationId}.`,
    );
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAllocationID.NotFound"
    )

```



```
    ) {
        console.warn(
            `${caught.message}. Did you provide the ID of a valid Elastic IP address
AllocationId?`,
        );
    } else {
        throw caught;
    }
}
};
```

- API-Einheiten finden Sie [AssociateAddress](#) unter AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
suspend fun associateAddressSc(
    instanceIdVal: String?,
    allocationIdVal: String?,
): String? {
    val associateRequest =
        AssociateAddressRequest {
            instanceId = instanceIdVal
            allocationId = allocationIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val associateResponse = ec2.associateAddress(associateRequest)
        return associateResponse.associationId
    }
}
```

- API-Einheiten finden Sie [AssociateAddress](#) in der AWS SDK-API-Kotlin-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Elastic IP-Adresse der angegebenen Instance in eine zugeordnete VPC zugeordnet.

```
C:\> Register-EC2Address -InstanceId i-12345678 -AllocationId eipalloc-12345678
```

Ausgabe:

```
eipassoc-12345678
```

Beispiel 2: In diesem Beispiel wird die angegebene Elastic IP-Adresse der angegebenen Instance in EC2 -Classic zugeordnet.

```
C:\> Register-EC2Address -InstanceId i-12345678 -PublicIp 203.0.113.17
```

- API-Einheiten finden Sie unter [AssociateAddress AWS Tools for PowerShell Cmdlet](#)-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
    actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""
```

```

    def __init__(
        self, allocation_id: str, public_ip: str, instance_id: Optional[str]
= None
    ) -> None:
        """
        Initializes the ElasticIp object.

        :param allocation_id: The allocation ID of the Elastic IP.
        :param public_ip: The public IP address of the Elastic IP.
        :param instance_id: The ID of the associated EC2 instance, if any.
        """
        self.allocation_id = allocation_id
        self.public_ip = public_ip
        self.instance_id = instance_id

def __init__(self, ec2_client: Any) -> None:
    """
    Initializes the ElasticIpWrapper with an EC2 client.

    :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                        access to AWS EC2 services.
    """
    self.ec2_client = ec2_client
    self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def associate(
        self, allocation_id: str, instance_id: str
    ) -> Union[Dict[str, Any], None]:
        """

```

```
Associates an Elastic IP address with an instance. When this association
is
created, the Elastic IP's public IP address is immediately used as the
public
IP address of the associated instance.

:param allocation_id: The allocation ID of the Elastic IP.
:param instance_id: The ID of the Amazon EC2 instance.
:return: A response that contains the ID of the association, or None if
no Elastic IP is found.
:raises ClientError: If the association fails, such as when the instance
ID is not found.
"""
    elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
    if elastic_ip is None:
        logger.info(f"No Elastic IP found with allocation ID
{allocation_id}.")
        return None

    try:
        response = self.ec2_client.associate_address(
            AllocationId=allocation_id, InstanceId=instance_id
        )
        elastic_ip.instance_id = (
            instance_id # Track the instance associated with this Elastic
IP.
        )
    except ClientError as err:
        if err.response["Error"]["Code"] == "InvalidInstanceID.NotFound":
            logger.error(
                f"Failed to associate Elastic IP {allocation_id} with
{instance_id} "
                "because the specified instance ID does not exist or has not
propagated fully. "
                "Verify the instance ID and try again, or wait a few moments
before attempting to "
                "associate the Elastic IP address."
            )
            raise
    return response
```

- API-Einheiten finden Sie unter [AssociateAddress](#) Python (Boto3) API -Referenz. AWS SDK

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
# Associates an Elastic IP address with an Amazon Elastic Compute Cloud
# (Amazon EC2) instance.
#
# Prerequisites:
#
# - The allocation ID corresponding to the Elastic IP address.
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @param instance_id [String] The ID of the instance.
# @return [String] The association ID corresponding to the association of the
#   Elastic IP address to the instance.
# @example
#   puts allocate_elastic_ip_address(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX',
#     'i-033c48ef067af3dEX')
def associate_elastic_ip_address_with_instance(
  ec2_client,
  allocation_id,
  instance_id
)
  response = ec2_client.associate_address(
    allocation_id: allocation_id,
    instance_id: instance_id
  )
  response.association_id
rescue StandardError => e
```

```
puts "Error associating Elastic IP address with instance: #{e.message}"  
'Error'  
end
```

- API-Einheiten finden Sie [AssociateAddress](#) unter AWS SDK for Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
pub async fn associate_ip_address(  
    &self,  
    allocation_id: &str,  
    instance_id: &str,  
) -> Result<AssociateAddressOutput, EC2Error> {  
    let response = self  
        .client  
        .associate_address()  
        .allocation_id(allocation_id)  
        .instance_id(instance_id)  
        .send()  
        .await?;  
    Ok(response)  
}
```

- API-Einheiten finden Sie [AssociateAddress](#) in der AWS SDK API Rust-Referenz.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
TRY.
    oo_result = lo_ec2->associateaddress(                                " oo_result
is returned for testing purposes. "
    iv_allocationid = iv_allocation_id
    iv_instanceid = iv_instance_id
    ).
    MESSAGE 'Associated an Elastic IP address with an EC2 instance.' TYPE
'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- API Einzelheiten finden Sie [AssociateAddress](#) unter AWS SDKSAPABAPAPI als Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **AssociateDhcpOptions** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `AssociateDhcpOptions`.

CLI

AWS CLI

Um einen DHCP Optionssatz mit Ihrem zu verknüpfen VPC

In diesem Beispiel wird der angegebene DHCP Optionssatz dem angegebenen zugewiesenen VPC. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 associate-dhcp-options --dhcp-options-id dopt-d9070ebb --vpc-id vpc-a01106c2
```

Um den DHCP Standardoptionssatz mit Ihrem zu verknüpfen VPC

In diesem Beispiel wird der DHCP Standardoptionssatz den angegebenen Optionen zugeordnet VPC. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 associate-dhcp-options --dhcp-options-id default --vpc-id vpc-a01106c2
```

- API Einzelheiten finden Sie [AssociateDhcpOptions](#) unter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel verknüpft den angegebenen DHCP Optionssatz mit dem angegebenen VPC.

```
Register-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d -VpcId vpc-12345678
```

Beispiel 2: In diesem Beispiel wird der DHCP Standardoptionssatz dem angegebenen Wert zugeordnet VPC.

```
Register-EC2DhcpOption -DhcpOptionsId default -VpcId vpc-12345678
```

- API Einzelheiten finden Sie unter [AssociateDhcpOptions AWS Tools for PowerShell Cmdlet](#)-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter. [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **AssociateRouteTable** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `AssociateRouteTable`.

CLI

AWS CLI

Um eine Routing-Tabelle einem Subnetz zuzuordnen

In diesem Beispiel wird die angegebene Routing-Tabelle dem angegebenen Subnetz zugeordnet.

Befehl:

```
aws ec2 associate-route-table --route-table-id rtb-22574640 --subnet-id subnet-9d4a7b6c
```

Ausgabe:

```
{  
  "AssociationId": "rtbassoc-781d0d1a"  
}
```

- API-Einheiten finden Sie unter [AssociateRouteTable AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Routing-Tabelle dem angegebenen Subnetz zugeordnet.

```
Register-EC2RouteTable -RouteTableId rtb-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

Ausgabe:

```
rtbassoc-12345678
```

- API-Einheiten finden Sie unter [AssociateRouteTable AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **AttachInternetGateway** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `AttachInternetGateway`.

CLI

AWS CLI

Um ein Internet-Gateway an Ihr anzuschließen VPC

Im folgenden `attach-internet-gateway` Beispiel wird das angegebene Internet-Gateway an das spezifische VPC angehängt.

```
aws ec2 attach-internet-gateway \  
  --internet-gateway-id igw-0d0fb496b3EXAMPLE \  
  --vpc-id vpc-0a60eb65b4EXAMPLE
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Internet-Gateways](#) im VPC Amazon-Benutzerhandbuch.

- API Einzelheiten finden Sie [AttachInternetGateway](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Internet-Gateway mit dem angegebenen VPC verbunden.

```
Add-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

Beispiel 2: In diesem Beispiel werden ein VPC und ein Internet-Gateway erstellt und anschließend das Internet-Gateway mit dem verbunden. VPC

```
$vpc = New-EC2Vpc -CidrBlock 10.0.0.0/16  
New-EC2InternetGateway | Add-EC2InternetGateway -VpcId $vpc.VpcId
```

- API-Einheiten finden Sie unter [AttachInternetGateway AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **AttachNetworkInterface** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `AttachNetworkInterface`.

CLI

AWS CLI

Beispiel 1: Um eine Netzwerkschnittstelle an eine Instanz anzuhängen

Im folgenden `attach-network-interface` Beispiel wird die angegebene Netzwerkschnittstelle an die angegebene Instanz angehängt.

```
aws ec2 attach-network-interface \  
  --network-interface-id eni-0dc56a8d4640ad10a \  
  --instance-id i-1234567890abcdef0 \  
  --device-index 1
```

Ausgabe:

```
{  
  "AttachmentId": "eni-attach-01a8fc87363f07cf9"  
}
```

Weitere Informationen finden Sie unter [Elastic Network Interfaces](#) im EC2 Amazon-Benutzerhandbuch.

Beispiel 2: Um eine Netzwerkschnittstelle an eine Instance mit mehreren Netzwerkkarten anzuschließen

Im folgenden `attach-network-interface` Beispiel wird die angegebene Netzwerkschnittstelle an die angegebene Instanz und Netzwerkkarte angehängt.

```
aws ec2 attach-network-interface \  
  --network-interface-id eni-0dc56a8d4640ad10a \  
  --instance-id i-1234567890abcdef0 \  
  --device-index 1
```

```
--network-interface-id eni-07483b1897541ad83 \  
--instance-id i-01234567890abcdef \  
--network-card-index 1 \  
--device-index 1
```

Ausgabe:

```
{  
  "AttachmentId": "eni-attach-0fbd7ee87a88cd06c"  
}
```

Weitere Informationen finden Sie unter [Elastic Network Interfaces](#) im EC2Amazon-Benutzerhandbuch.

- APIEinzelheiten finden Sie [AttachNetworkInterface](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle an die angegebene Instanz angehängt.

```
Add-EC2NetworkInterface -NetworkInterfaceId eni-12345678 -InstanceId i-1a2b3c4d -  
DeviceIndex 1
```

Ausgabe:

```
eni-attach-1a2b3c4d
```

- APIEinzelheiten finden Sie unter [AttachNetworkInterface AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **AttachVolume** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `AttachVolume`.

CLI

AWS CLI

Um ein Volume an eine Instanz anzuhängen

Mit diesem Beispielbefehl wird ein Volume (`vol-1234567890abcdef0`) an eine Instanz (`i-01474ef662b89480`) angehängt als `/dev/sdf`.

Befehl:

```
aws ec2 attach-volume --volume-id vol-1234567890abcdef0 --instance-id i-01474ef662b89480 --device /dev/sdf
```

Ausgabe:

```
{
  "AttachTime": "YYYY-MM-DDTHH:MM:SS.000Z",
  "InstanceId": "i-01474ef662b89480",
  "VolumeId": "vol-1234567890abcdef0",
  "State": "attaching",
  "Device": "/dev/sdf"
}
```

- API-Einheiten finden Sie [AttachVolume](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Volume an die angegebene Instanz angehängt und mit dem angegebenen Gerätenamen verfügbar gemacht.

```
Add-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

Ausgabe:

```
AttachTime           : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
```

```
Device           : /dev/sdh
InstanceId        : i-1a2b3c4d
State            : attaching
VolumeId         : vol-12345678
```

- API-Einheiten finden Sie unter [AttachVolume AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **AttachVpnGateway** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `AttachVpnGateway`.

CLI

AWS CLI

Um ein virtuelles privates Gateway an Ihr anzuhängen VPC

Im folgenden `attach-vpn-gateway` Beispiel wird das angegebene virtuelle private Gateway an das angegebene VPC angehängt.

```
aws ec2 attach-vpn-gateway \
  --vpn-gateway-id vgw-9a4cacf3 \
  --vpc-id vpc-a01106c2
```

Ausgabe:

```
{
  "VpcAttachment": {
    "State": "attaching",
    "VpcId": "vpc-a01106c2"
  }
}
```

- API-Einheiten finden Sie unter [AttachVpnGateway AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene virtuelle private Gateway an das angegebene VPC angehängt.

```
Add-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

Ausgabe:

```
State      VpcId
-----
attaching  vpc-12345678
```

- API-Einheiten finden Sie unter [AttachVpnGateway AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **AuthorizeSecurityGroupEgress** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung von `AuthorizeSecurityGroupEgress`.

CLI

AWS CLI

Um eine Regel hinzuzufügen, die ausgehenden Verkehr in einen bestimmten Adressbereich zulässt

Dieser Beispielbefehl fügt eine Regel hinzu, die Zugriff auf die angegebenen Adressbereiche an TCP Port 80 gewährt.

Befehl (Linux):

```
aws ec2 authorize-security-group-egress --group-id sg-1a2b3c4d --ip-permissions IpProtocol=tcp,FromPort=80,ToPort=80,IpRanges='[{"CidrIp=10.0.0.0/16}]'
```

Befehl (Windows):

```
aws ec2 authorize-security-group-egress --group-id sg-1a2b3c4d --ip-
permissions IpProtocol=tcp,FromPort=80,ToPort=80,IpRanges=[{CidrIp=10.0.0.0/16}]
```

Um eine Regel hinzuzufügen, die ausgehenden Datenverkehr zu einer bestimmten Sicherheitsgruppe zulässt

Dieser Beispielbefehl fügt eine Regel hinzu, die Zugriff auf die angegebene Sicherheitsgruppe an TCP Port 80 gewährt.

Befehl (Linux):

```
aws ec2 authorize-security-group-egress --group-id sg-1a2b3c4d --ip-permissions
IpProtocol=tcp,FromPort=80,ToPort=80,UserIdGroupPairs='[{GroupId=sg-4b51a32f}]'
```

Befehl (Windows):

```
aws ec2 authorize-security-group-egress --group-id sg-1a2b3c4d --ip-
permissions IpProtocol=tcp,FromPort=80,ToPort=80,UserIdGroupPairs=[{GroupId=sg-4b51a32f}]
```

- API-Einheiten finden Sie [AuthorizeSecurityGroupEgress](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel definiert eine Ausgangsregel für die angegebene Sicherheitsgruppe für EC2 -VPC. Die Regel gewährt Zugriff auf den angegebenen IP-Adressbereich an TCP Port 80. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80";
  IpRanges="203.0.113.0/24" }
Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um das Objekt zu erstellen. IpPermission

```
$ip = New-Object Amazon.EC2.Model.IpPermission
```



```
$ip.IpProtocol = "tcp"
$ip.FromPort = 80
$ip.ToPort = 80
$ip.IpRanges.Add("203.0.113.0/24")

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 3: Dieses Beispiel gewährt Zugriff auf die angegebene Quellsicherheitsgruppe auf TCP Port 80.

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission
@( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- API-Einheiten finden Sie unter [AuthorizeSecurityGroupEgress AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie es **AuthorizeSecurityGroupIngress** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `AuthorizeSecurityGroupIngress`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Authorize the local computer ingress to EC2 instances associated
/// with the virtual private cloud (VPC) security group.
/// </summary>
/// <param name="groupName">The name of the security group.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> AuthorizeSecurityGroupIngress(string groupName)
{
    try
    {
        // Get the IP address for the local computer.
        var ipAddress = await GetIpAddress();
        Console.WriteLine($"Your IP address is: {ipAddress}");
        var ipRanges =
            new List<IpRange> { new IpRange { CidrIp = $"{ipAddress}/32" } };
        var permission = new IpPermission
        {
            Ipv4Ranges = ipRanges,
            IpProtocol = "tcp",
            FromPort = 22,
            ToPort = 22
        };
        var permissions = new List<IpPermission> { permission };
        var response = await _amazonEC2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest(groupName,
permissions));
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidPermission.Duplicate")
        {
```

```
        _logger.LogError(
            $"The ingress rule already exists. {ec2Exception.Message}");
    }

    throw;
}
catch (Exception ex)
{
    _logger.LogError(
        $"An error occurred while authorizing ingress.: {ex.Message}");
    throw;
}
}


/// <summary>
/// Authorize the local computer for ingress to
/// the Amazon EC2 SecurityGroup.
/// </summary>
/// <returns>The IPv4 address of the computer running the scenario.</returns>
private static async Task<string> GetIpAddress()
{
    var httpClient = new HttpClient();
    var ipString = await httpClient.GetStringAsync("https://
checkip.amazonaws.com");

    // The IP address is returned with a new line
    // character on the end. Trim off the whitespace and
    // return the value to the caller.
    return ipString.Trim();
}
```

- API-Einheiten finden Sie [AuthorizeSecurityGroupIngress](#) unter AWS SDK for .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

 Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#####
# function ec2_authorize_security_group_ingress
#
# This function authorizes an ingress rule for an Amazon Elastic Compute Cloud
  (Amazon EC2) security group.
#
# Parameters:
#   -g security_group_id - The ID of the security group.
#   -i ip_address - The IP address or CIDR block to authorize.
#   -p protocol - The protocol to authorize (e.g., tcp, udp, icmp).
#   -f from_port - The start of the port range to authorize.
#   -t to_port - The end of the port range to authorize.
#
# And:
#   0 - If successful.
#   1 - If it fails.
#####
function ec2_authorize_security_group_ingress() {
    local security_group_id ip_address protocol from_port to_port response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_authorize_security_group_ingress"
        echo "Authorizes an ingress rule for an Amazon Elastic Compute Cloud (Amazon
        EC2) security group."
        echo "  -g security_group_id - The ID of the security group."
        echo "  -i ip_address - The IP address or CIDR block to authorize."
        echo "  -p protocol - The protocol to authorize (e.g., tcp, udp, icmp)."
        echo "  -f from_port - The start of the port range to authorize."
        echo "  -t to_port - The end of the port range to authorize."
        echo ""
    }
}
```

```
}

# Retrieve the calling parameters.
while getopts "g:i:p:f:t:h" option; do
  case "${option}" in
    g) security_group_id="${OPTARG}" ;;
    i) ip_address="${OPTARG}" ;;
    p) protocol="${OPTARG}" ;;
    f) from_port="${OPTARG}" ;;
    t) to_port="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$security_group_id" ]]; then
  errecho "ERROR: You must provide a security group ID with the -g parameter."
  usage
  return 1
fi

if [[ -z "$ip_address" ]]; then
  errecho "ERROR: You must provide an IP address or CIDR block with the -i
parameter."
  usage
  return 1
fi

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

if [[ -z "$from_port" ]]; then
  errecho "ERROR: You must provide a start port with the -f parameter."
```

```

    usage
    return 1
fi

if [[ -z "$to_port" ]]; then
    errecho "ERROR: You must provide an end port with the -t parameter."
    usage
    return 1
fi

response=$(aws ec2 authorize-security-group-ingress \
  --group-id "$security_group_id" \
  --cidr "${ip_address}/32" \
  --protocol "$protocol" \
  --port "$from_port-$to_port" \
  --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports authorize-security-group-ingress operation
failed.$response"
    return 1
}

return 0
}

```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:

```

```

#      $1 - The error code returned by the AWS CLI.
#
# Returns:
#      0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then
        errecho " The service returned an error."
    elif [ "$err_code" == 255 ]; then
        errecho " 255 is a catch-all error."
    fi

    return 0
}

```

- API-Einheiten finden Sie [AuthorizeSecurityGroupIngress](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#!/usr/bin/env bash
//! Authorize ingress to an Amazon Elastic Compute Cloud (Amazon EC2) group.
```

```

/ * !
  \param groupID: The EC2 group ID.
  \param clientConfiguration: The ClientConfiguration object.
  \return bool: True if the operation was successful, false otherwise.
  * /
bool
AwsDoc::EC2::authorizeSecurityGroupIngress(const Aws::String &groupID,
                                           const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::AuthorizeSecurityGroupIngressRequest
authorizeSecurityGroupIngressRequest;
    authorizeSecurityGroupIngressRequest.SetGroupId(groupID);
    buildSampleIngressRule(authorizeSecurityGroupIngressRequest);

    Aws::EC2::Model::AuthorizeSecurityGroupIngressOutcome
authorizeSecurityGroupIngressOutcome =

ec2Client.AuthorizeSecurityGroupIngress(authorizeSecurityGroupIngressRequest);

    if (authorizeSecurityGroupIngressOutcome.IsSuccess()) {
        std::cout << "Successfully authorized security group ingress." <<
std::endl;
    } else {
        std::cerr << "Error authorizing security group ingress: "
<< authorizeSecurityGroupIngressOutcome.GetError().GetMessage()
<< std::endl;
    }

    return authorizeSecurityGroupIngressOutcome.IsSuccess();
}

```

Utility-Funktion zum Erstellen einer Eingangsregel.

```

// ! Build a sample ingress rule.
/ * !
  \param authorize_request: An 'AuthorizeSecurityGroupIngressRequest' instance.
  \return void:
  * /
void buildSampleIngressRule(
    Aws::EC2::Model::AuthorizeSecurityGroupIngressRequest &authorize_request)
{

```



```
Aws::String ingressIPRange = "203.0.113.0/24"; // Configure this for your
allowed IP range.
Aws::EC2::Model::IpRange ip_range;
ip_range.SetCidrIp(ingressIPRange);

Aws::EC2::Model::IpPermission permission1;
permission1.SetIpProtocol("tcp");
permission1.SetToPort(80);
permission1.SetFromPort(80);
permission1.AddIpRanges(ip_range);

authorize_request.AddIpPermissions(permission1);

Aws::EC2::Model::IpPermission permission2;
permission2.SetIpProtocol("tcp");
permission2.SetToPort(22);
permission2.SetFromPort(22);
permission2.AddIpRanges(ip_range);

authorize_request.AddIpPermissions(permission2);
}
```

- API-Einheiten finden Sie unter [AuthorizeSecurityGroupIngress AWS SDK for C++ API-Referenz](#).

CLI

AWS CLI

Beispiel 1: Um eine Regel hinzuzufügen, die eingehenden Datenverkehr SSH zulässt

Das folgende `authorize-security-group-ingress` Beispiel fügt eine Regel hinzu, die eingehenden Verkehr auf TCP Port 22 () SSH zulässt.

```
aws ec2 authorize-security-group-ingress \
  --group-id sg-1234567890abcdef0 \
  --protocol tcp \
  --port 22 \
  --cidr 203.0.113.0/24
```

Ausgabe:

```
{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "IsEgress": false,
      "IpProtocol": "tcp",
      "FromPort": 22,
      "ToPort": 22,
      "CidrIpv4": "203.0.113.0/24"
    }
  ]
}
```

Beispiel 2: Um eine Regel hinzuzufügen, die eingehenden HTTP Verkehr von einer anderen Sicherheitsgruppe zulässt

Im folgenden `authorize-security-group-ingress` Beispiel wird eine Regel hinzugefügt, die eingehenden Zugriff auf TCP Port 80 von der Quellsicherheitsgruppe aus ermöglicht. `sg-1a2b3c4d` Die Quellgruppe muss sich in derselben Gruppe VPC oder in einem Peer befinden VPC (erfordert eine VPC Peering-Verbindung). Eingehender Datenverkehr ist basierend auf den privaten IP-Adressen der Instances erlaubt, die der Quellsicherheitsgruppe zugeordnet sind (nicht die öffentliche IP-Adresse oder die Elastic-IP-Adresse).

```
aws ec2 authorize-security-group-ingress \
  --group-id sg-1234567890abcdef0 \
  --protocol tcp \
  --port 80 \
  --source-group sg-1a2b3c4d
```

Ausgabe:

```
{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "SecurityGroupRuleId": "sgr-01f4be99110f638a7",
    }
  ]
}
```

```

        "IsEgress": false,
        "IpProtocol": "tcp",
        "FromPort": 80,
        "ToPort": 80,
        "ReferencedGroupInfo": {
            "GroupId": "sg-1a2b3c4d",
            "UserId": "123456789012"
        }
    }
]
}

```

Beispiel 3: So fügen Sie mehrere Regeln im selben Aufruf hinzu

Im folgenden `authorize-security-group-ingress` Beispiel werden mithilfe des `ip-permissions` Parameters zwei Regeln für eingehenden Datenverkehr hinzugefügt, von denen eine den eingehenden Zugriff auf TCP Port 3389 (RDP) und die andere, die ping/aktiviert. ICMP

```
aws ec2 authorize-security-group-ingress --group-id sg-1234567890abcdef0 --ip-permissions
IpProtocol =tcp, =3389, FromPort ToPort =3389, IpRanges = "[{CidrIp=172.31.0.0/16}]"
IpProtocol =icmp, =-1, FromPort ToPort =-1, IpRanges = "[{CidrIp=172.31.0.0/16}]"
```

Ausgabe:

```

{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupRuleId": "sgr-00e06e5d3690f29f3",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "IsEgress": false,
      "IpProtocol": "tcp",
      "FromPort": 3389,
      "ToPort": 3389,
      "CidrIpv4": "172.31.0.0/16"
    },
    {
      "SecurityGroupRuleId": "sgr-0a133dd4493944b87",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",

```

```

        "IsEgress": false,
        "IpProtocol": "tcp",
        "FromPort": -1,
        "ToPort": -1,
        "CidrIpv4": "172.31.0.0/16"
    }
]
}

```

Beispiel 4: Um eine Regel ICMP für den Datenverkehr hinzuzufügen

Im folgenden `authorize-security-group-ingress` Beispiel wird der `ip-permissions` Parameter verwendet, um eine Regel für eingehende Nachrichten hinzuzufügen, die die ICMP Nachricht `Destination Unreachable: Fragmentation Needed and Don't Fragment was Set` (Typ 3, Code 4) von überall aus zulässt.

```
aws ec2 authorize-security-group-ingress --group-id sg-1234567890abcdef0 --ip-permissions =icmp, =3, =4, = [{"=0.0.0.0/0}] IpProtocol FromPort ToPort IpRanges CidrIp
```

Ausgabe:

```

{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupRuleId": "sgr-0de3811019069b787",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "IsEgress": false,
      "IpProtocol": "icmp",
      "FromPort": 3,
      "ToPort": 4,
      "CidrIpv4": "0.0.0.0/0"
    }
  ]
}

```

Beispiel 5: Um eine Regel für Traffic IPv6 hinzuzufügen

Im folgenden `authorize-security-group-ingress` Beispiel wird der `ip-permissions` Parameter verwendet, um eine Regel für eingehenden Datenverkehr hinzuzufügen, die SSH den Zugriff (Port 22) aus dem IPv6 Bereich `2001:db8:1234:1a00::/64` ermöglicht.

```
aws ec2 authorize-security-group-ingress --group-id sg-1234567890abcdef0 --ip-permissions
IpProtocol =tcp, =22, =22, Ipv6Ranges= "[{6=2001:db 8:1234:1 a00: :/64}]" FromPort ToPort
CidrIpv6
```

Ausgabe:

```
{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupRuleId": "sgr-0455bc68b60805563",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "IsEgress": false,
      "IpProtocol": "tcp",
      "FromPort": 22,
      "ToPort": 22,
      "CidrIpv6": "2001:db8:1234:1a00::/64"
    }
  ]
}
```

Beispiel 6: Um eine Regel für ICMPv6 Traffic hinzuzufügen

Im folgenden `authorize-security-group-ingress` Beispiel wird der `ip-permissions` Parameter verwendet, um eine Regel für eingehenden Datenverkehr hinzuzufügen, die ICMPv6 Datenverkehr von überall her zulässt.

```
aws ec2 authorize-security-group-ingress --group-id sg-1234567890abcdef0 --ip-permissions
=icmpv6, Ipv6Ranges= "[{6=: :/0}]" IpProtocol CidrIpv6
```

Ausgabe:

```
{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupRuleId": "sgr-04b612d9363ab6327",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "IsEgress": false,
      "IpProtocol": "icmpv6",

```

```

        "FromPort": -1,
        "ToPort": -1,
        "CidrIpv6": ":::/0"
    }
]
}

```

Beispiel 7: Eine Regel mit einer Beschreibung hinzufügen

Im folgenden `authorize-security-group-ingress` Beispiel wird der Parameter `ip-permissions` verwendet, um eine Regel für `ip-permissions` eingehenden Datenverkehr hinzuzufügen, die den Datenverkehr aus dem angegebenen Adressbereich zulässt. IPv4 Die Regel enthält eine Beschreibung, die Ihnen später hilft, sie zu identifizieren.

```
aws ec2 authorize-security-group-ingress --group-id sg-1234567890abcdef0 --ip-permissions
IpProtocol=tcp,=3389,=3389,="[=203.0.113.0/24, Description='Zugriff von einer
Niederlassung in New York aus']" ToPort IpRanges CidrIp RDP
```

Ausgabe:

```

{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupId": "sgr-0397bbcc01e974db3",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "IsEgress": false,
      "IpProtocol": "tcp",
      "FromPort": 3389,
      "ToPort": 3389,
      "CidrIpv4": "203.0.113.0/24",
      "Description": "RDP access from NY office"
    }
  ]
}

```

Beispiel 8: So fügen Sie eine eingehende Regel hinzu, die eine Präfixliste verwendet

Im folgenden `authorize-security-group-ingress` Beispiel wird der Parameter `ip-permissions` verwendet, um eine Regel für `ip-permissions` eingehenden Datenverkehr hinzuzufügen, die den gesamten Verkehr für die Bereiche in der angegebenen CIDR Präfixliste zulässt.

```
aws ec2 authorize-security-group-ingress --group-id sg-04a351bfe432d4e71 --ip-permissions
=all, = [{"=pl-002dc3ec097de1514}]" IpProtocol PrefixListIds PrefixListId
```

Ausgabe:

```
{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupId": "sgr-09c74b32f677c6c7c",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "IsEgress": false,
      "IpProtocol": "-1",
      "FromPort": -1,
      "ToPort": -1,
      "PrefixListId": "pl-0721453c7ac4ec009"
    }
  ]
}
```

Weitere Informationen finden Sie unter [Sicherheitsgruppen](#) im VPCAmazon-Benutzerhandbuch.

- API-Einheiten finden Sie [AuthorizeSecurityGroupIngress](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Creates a new security group asynchronously with the specified group name,
 * description, and VPC ID. It also
 * authorizes inbound traffic on ports 80 and 22 from the specified IP
 * address.
```

```
*
* @param groupName    the name of the security group to create
* @param groupDesc    the description of the security group
* @param vpcId        the ID of the VPC in which to create the security
group
* @param myIpAddress  the IP address from which to allow inbound traffic
(e.g., "192.168.1.1/0" to allow traffic from
*                      any IP address in the 192.168.1.0/24 subnet)
* @return a CompletableFuture that, when completed, returns the ID of the
created security group
* @throws RuntimeException if there was a failure creating the security
group or authorizing the inbound traffic
*/
public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
    CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
        .groupName(groupName)
        .description(groupDesc)
        .vpcId(vpcId)
        .build();

    return getAsyncClient().createSecurityGroup(createRequest)
        .thenCompose(createResponse -> {
            String groupId = createResponse.groupId();
            IpRange ipRange = IpRange.builder()
                .cidrIp(myIpAddress + "/32")
                .build();

            IpPermission ipPerm = IpPermission.builder()
                .ipProtocol("tcp")
                .toPort(80)
                .fromPort(80)
                .ipRanges(ipRange)
                .build();

            IpPermission ipPerm2 = IpPermission.builder()
                .ipProtocol("tcp")
                .toPort(22)
                .fromPort(22)
                .ipRanges(ipRange)
                .build();
```



```

        AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
    .groupName(groupName)
    .ipPermissions(ipPerm, ipPerm2)
    .build();

        return
getAsyncClient().authorizeSecurityGroupIngress(authRequest)
    .thenApply(authResponse -> groupId);
    })
    .whenComplete((result, exception) -> {
        if (exception != null) {
            if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                throw (Ec2Exception) exception.getCause();
            } else {
                throw new RuntimeException("Failed to create security
group: " + exception.getMessage(), exception);
            }
        }
    });
}

```

- API Einheiten finden Sie [AuthorizeSecurityGroupIngress](#) unter AWS SDK for Java 2.x API Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import {
    AuthorizeSecurityGroupIngressCommand,
    EC2Client,
} from "@aws-sdk/client-ec2";

```


```
/**
 * Adds the specified inbound (ingress) rules to a security group.
 * @param {{ groupId: string, ipAddress: string }} options
 */
export const main = async ({ groupId, ipAddress }) => {
  const client = new EC2Client({});
  const command = new AuthorizeSecurityGroupIngressCommand({
    // Use a group ID from the AWS console or
    // the DescribeSecurityGroupsCommand.
    GroupId: groupId,
    IpPermissions: [
      {
        IpProtocol: "tcp",
        FromPort: 22,
        ToPort: 22,
        // The IP address to authorize.
        // For more information on this notation, see
        // https://en.wikipedia.org/wiki/Classless_Inter-
Domain_Routing#CIDR_notation
        IpRanges: [{ CidrIp: `${ipAddress}/32` }],
      },
    ],
  });

  try {
    const { SecurityGroupRules } = await client.send(command);
    console.log(JSON.stringify(SecurityGroupRules, null, 2));
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
      console.warn(`${caught.message}. Please provide a valid GroupId.`);
    } else {
      throw caught;
    }
  }
};
```

- API Einzelheiten finden Sie [AuthorizeSecurityGroupIngress](#) unter AWS SDK for JavaScript API Referenz.

Kotlin

SDK für Kotlin

 Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
suspend fun createEC2SecurityGroupSc(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
    myIpAddress: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
        val ipRange =
            IpRange {
                cidrIp = "$myIpAddress/0"
            }

        val ipPerm =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 80
                fromPort = 80
                ipRanges = listOf(ipRange)
            }

        val ipPerm2 =
            IpPermission {
                ipProtocol = "tcp"
                toPort = 22
            }
    }
```

```
        fromPort = 22
        ipRanges = listOf(ipRange)
    }

    val authRequest =
        AuthorizeSecurityGroupIngressRequest {
            groupName = groupNameVal
            ipPermissions = listOf(ipPerm, ipPerm2)
        }
    ec2.authorizeSecurityGroupIngress(authRequest)
    println("Successfully added ingress policy to Security Group
    $groupNameVal")
    return resp.groupId
}
}
```

- API-Einheiten finden Sie [AuthorizeSecurityGroupIngress](#) in der AWS SDK-API-Kotlin-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden Eingangsregeln für eine Sicherheitsgruppe für EC2 - VPC definiert. Diese Regeln gewähren Zugriff auf eine bestimmte IP-Adresse für SSH (Port 22) und RDC (Port 3389). Beachten Sie, dass Sie Sicherheitsgruppen für identifizieren müssen EC2 — und zwar VPC anhand der Sicherheitsgruppen-ID, nicht anhand des Sicherheitsgruppennamens. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip1 = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
  IpRanges="203.0.113.25/32" }
$ip2 = @{ IpProtocol="tcp"; FromPort="3389"; ToPort="3389";
  IpRanges="203.0.113.25/32" }

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission @( $ip1, $ip2 )
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um die IpPermission Objekte zu erstellen.

```
$ip1 = New-Object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")

$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission @( $ip1, $ip2 )
```

Beispiel 3: In diesem Beispiel werden Eingangsregeln für eine Sicherheitsgruppe für -Classic definiert. EC2 Diese Regeln gewähren Zugriff auf eine bestimmte IP-Adresse für SSH (Port 22) und RDC (Port 3389). Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip1 = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
  IpRanges="203.0.113.25/32" }
$ip2 = @{ IpProtocol="tcp"; FromPort="3389"; ToPort="3389";
  IpRanges="203.0.113.25/32" }

Grant-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission
@( $ip1, $ip2 )
```

Beispiel 4: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um die IpPermission Objekte zu erstellen.

```
$ip1 = New-Object Amazon.EC2.Model.IpPermission
$ip1.IpProtocol = "tcp"
$ip1.FromPort = 22
$ip1.ToPort = 22
$ip1.IpRanges.Add("203.0.113.25/32")

$ip2 = new-object Amazon.EC2.Model.IpPermission
$ip2.IpProtocol = "tcp"
$ip2.FromPort = 3389
$ip2.ToPort = 3389
$ip2.IpRanges.Add("203.0.113.25/32")
```

```
Grant-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission
@( $ip1, $ip2 )
```

Beispiel 5: Dieses Beispiel gewährt TCP Port 8081 Zugriff von der angegebenen Quellsicherheitsgruppe (sg-1a2b3c4d) auf die angegebene Sicherheitsgruppe (sg-12345678).

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair
$ug.GroupId = "sg-1a2b3c4d"
$ug.UserId = "123456789012"

Grant-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission
@( @{ IpProtocol="tcp"; FromPort="8081"; ToPort="8081"; UserIdGroupPairs=$ug } )
```

Beispiel 6: In diesem Beispiel wird CIDR 5.5.5.5/32 zu den Eingangsregeln der Sicherheitsgruppe sg-1234abcd für Port 22-Verkehr mit einer Beschreibung hinzugefügt. TCP

```
$IpRange = New-Object -TypeName Amazon.EC2.Model.IpRange
$IpRange.CidrIp = "5.5.5.5/32"
$IpRange.Description = "SSH from Office"
$IpPermission = New-Object Amazon.EC2.Model.IpPermission
$IpPermission.IpProtocol = "tcp"
$IpPermission.ToPort = 22
$IpPermission.FromPort = 22
$IpPermission.Ipv4Ranges = $IpRange
Grant-EC2SecurityGroupIngress -GroupId sg-1234abcd -IpPermission $IpPermission
```

- Einzelheiten finden Sie unter Cmdlet-Referenz. API [AuthorizeSecurityGroupIngress](#) AWS Tools for PowerShell

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class SecurityGroupWrapper:
```

```
"""Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
actions."""

def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
None):
    """
    Initializes the SecurityGroupWrapper with an EC2 client and an optional
    security group ID.

    :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                        access to AWS EC2 services.
    :param security_group: The ID of a security group to manage. This is a
high-level identifier
                        that represents the security group.
    """
    self.ec2_client = ec2_client
    self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def authorize_ingress(self, ssh_ingress_ip: str) -> Optional[Dict[str, Any]]:
        """
        Adds a rule to the security group to allow access to SSH.

        :param ssh_ingress_ip: The IP address that is granted inbound access to
connect
                                to port 22 over TCP, used for SSH.
        :return: The response to the authorization request. The 'Return' field of
the
                    response indicates whether the request succeeded or failed, or
None if no security group is set.
        :raise Handles AWS SDK service-level ClientError, with special handling
for ResourceAlreadyExists
        """
```

```
"""
if self.security_group is None:
    logger.info("No security group to update.")
    return None

try:
    ip_permissions = [
        {
            # SSH ingress open to only the specified IP address.
            "IpProtocol": "tcp",
            "FromPort": 22,
            "ToPort": 22,
            "IpRanges": [{"CidrIp": f"{ssh_ingress_ip}/32"}],
        }
    ]
    response = self.ec2_client.authorize_security_group_ingress(
        GroupId=self.security_group, IpPermissions=ip_permissions
    )
except ClientError as err:
    if err.response["Error"]["Code"] == "InvalidPermission.Duplicate":
        logger.error(
            f"The SSH ingress rule for IP {ssh_ingress_ip} already
exists"
            f"in security group '{self.security_group}'."
        )
        raise
    else:
        return response
```

- API-Einheiten finden Sie unter [AuthorizeSecurityGroupIngress](#) Python (Boto3) API - Referenz.AWS SDK

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// Add an ingress rule to a security group explicitly allowing IPv4 address
/// as {ip}/32 over TCP port 22.
pub async fn authorize_security_group_ssh_ingress(
    &self,
    group_id: &str,
    ingress_ips: Vec<Ipv4Addr>,
) -> Result<(), EC2Error> {
    tracing::info!("Authorizing ingress for security group {group_id}");
    self.client
        .authorize_security_group_ingress()
        .group_id(group_id)
        .set_ip_permissions(Some(
            ingress_ips
                .into_iter()
                .map(|ip| {
                    IpPermission::builder()
                        .ip_protocol("tcp")
                        .from_port(22)
                        .to_port(22)
                        .ip_ranges(IpRange::builder().cidr_ip(format!(
                            "{ip}/32"))).build())
                        .build()
                })
                .collect(),
        ))
        .send()
        .await?;
    Ok(())
}
```

- API-Einheiten finden Sie [AuthorizeSecurityGroupIngress](#) in der AWS SDK API-Rust-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie `CancelCapacityReservation` mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CancelCapacityReservation`.

CLI

AWS CLI

Um eine Kapazitätsreservierung zu stornieren

Im folgenden `cancel-capacity-reservation` Beispiel wird die angegebene Kapazitätsreservierung storniert.

```
aws ec2 cancel-capacity-reservation \  
  --capacity-reservation-id cr-1234abcd56EXAMPLE
```

Ausgabe:

```
{  
  "Return": true  
}
```

Weitere Informationen finden Sie unter [Stornieren einer Kapazitätsreservierung](#) im Amazon Elastic Compute Cloud-Benutzerhandbuch für Linux-Instances.

- API-Einheiten finden Sie [CancelCapacityReservation](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Kapazitätsreservierung `cr-0c1f2345db6f7cdba` storniert

```
Remove-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2CapacityReservation
(CancelCapacityReservation)" on target "cr-0c1f2345db6f7cdba".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): y
True
```

- API-Einheiten finden [CancelCapacityReservation AWS Tools for PowerShell](#) Sie unter Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **CancelImportTask** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CancelImportTask`.

CLI

AWS CLI

Um eine Importaufgabe abubrechen

Im folgenden `cancel-import-task` Beispiel wird die angegebene Aufgabe zum Importieren eines Bilds abgebrochen.

```
aws ec2 cancel-import-task \
  --import-task-id import-ami-1234567890abcdef0
```

Ausgabe:

```
{
  "ImportTaskId": "import-ami-1234567890abcdef0",
```

```
"PreviousState": "active",  
"State": "deleting"  
}
```

- API-Einheiten finden Sie [CancelImportTask](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Importaufgabe (entweder Snapshot- oder Bildimport) abgebrochen. Falls erforderlich, kann mithilfe des **-CancelReason** Parameters ein Grund angegeben werden.

```
Stop-EC2ImportTask -ImportTaskId import-ami-abcdefgh
```

- API-Einheiten finden Sie unter [CancelImportTask AWS Tools for PowerShell Cmdlet](#)-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **CancelSpotFleetRequests** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CancelSpotFleetRequests`.

CLI

AWS CLI

Beispiel 1: Um eine Spot-Flottenanfrage zu stornieren und die zugehörigen Instances zu beenden

Im folgenden `cancel-spot-fleet-requests` Beispiel wird eine Spot-Flottenanfrage storniert und die zugehörigen On-Demand-Instances und Spot-Instances beendet.

```
aws ec2 cancel-spot-fleet-requests \  
  --spot-fleet-request-ids sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE \  
  --cancel-reason Spot Instance terminated
```

--terminate-instances

Ausgabe:

```
{
  "SuccessfulFleetRequests": [
    {
      "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
      "CurrentSpotFleetRequestState": "cancelled_terminating",
      "PreviousSpotFleetRequestState": "active"
    }
  ],
  "UnsuccessfulFleetRequests": []
}
```

Weitere Informationen finden Sie unter [Stornieren einer Spot-Flottenanfrage](#) im Amazon Elastic Compute Cloud-Benutzerhandbuch für Linux-Instances.

Beispiel 2: Um eine Spot-Flottenanfrage zu stornieren, ohne die zugehörigen Instances zu beenden

Im folgenden `cancel-spot-fleet-requests` Beispiel wird eine Spot-Flottenanforderung storniert, ohne die zugehörigen On-Demand-Instances und Spot-Instances zu beenden.

```
aws ec2 cancel-spot-fleet-requests \
  --spot-fleet-request-ids sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE \
  --no-terminate-instances
```

Ausgabe:

```
{
  "SuccessfulFleetRequests": [
    {
      "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
      "CurrentSpotFleetRequestState": "cancelled_running",
      "PreviousSpotFleetRequestState": "active"
    }
  ],
  "UnsuccessfulFleetRequests": []
}
```

Weitere Informationen finden Sie unter [Stornieren einer Spot-Flottenanfrage](#) im Amazon Elastic Compute Cloud-Benutzerhandbuch für Linux-Instances.

- API-Einheiten finden Sie [CancelSpotFleetRequests](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Spot-Flottenanforderung storniert und die zugehörigen Spot-Instances beendet.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $true
```

Beispiel 2: In diesem Beispiel wird die angegebene Spot-Flottenanforderung storniert, ohne die zugehörigen Spot-Instances zu beenden.

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $false
```

- API-Einheiten finden Sie unter [CancelSpotFleetRequests AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **CancelSpotInstanceRequests** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CancelSpotInstanceRequests`.

CLI

AWS CLI

Um Spot-Instance-Anfragen zu stornieren

Mit diesem Beispielbefehl wird eine Spot-Instance-Anfrage storniert.

Befehl:

```
aws ec2 cancel-spot-instance-requests --spot-instance-request-ids sir-08b93456
```

Ausgabe:

```
{
  "CancelledSpotInstanceRequests": [
    {
      "State": "cancelled",
      "SpotInstanceRequestId": "sir-08b93456"
    }
  ]
}
```

- API-Einheiten finden Sie [CancelSpotInstanceRequests](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Spot-Instance-Anfrage storniert.

```
Stop-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Ausgabe:

SpotInstanceRequestId	State
-----	-----
sir-12345678	cancelled

- API-Einheiten finden Sie unter [CancelSpotInstanceRequests AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **ConfirmProductInstance** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `ConfirmProductInstance`.

CLI

AWS CLI

Um die Produktinstanz zu bestätigen

In diesem Beispiel wird ermittelt, ob der angegebene Produktcode der angegebenen Instanz zugeordnet ist.

Befehl:

```
aws ec2 confirm-product-instance --product-code 774F4FF8 --instance-id i-1234567890abcdef0
```

Ausgabe:

```
{  
  "OwnerId": "123456789012"  
}
```

- API-Einheiten finden Sie [ConfirmProductInstance](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ermittelt, ob der angegebene Produktcode der angegebenen Instanz zugeordnet ist.

```
Confirm-EC2ProductInstance -ProductCode 774F4FF8 -InstanceId i-12345678
```

- API-Einheiten finden Sie unter [ConfirmProductInstance AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **CopyImage** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CopyImage`.

CLI

AWS CLI

Beispiel 1: Um eine in eine andere Region AMI zu kopieren

Der folgende `copy-image` Beispielbefehl kopiert die angegebene Datei AMI aus der `us-west-2` Region in die `us-east-1` Region und fügt eine kurze Beschreibung hinzu.

```
aws ec2 copy-image \  
  --region us-east-1 \  
  --name ami-name \  
  --source-region us-west-2 \  
  --source-image-id ami-066877671789bd71b \  
  --description "This is my copied image."
```

Ausgabe:

```
{  
  "ImageId": "ami-0123456789abcdefg"  
}
```

Weitere Informationen finden Sie unter [Kopieren einer AMI](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 2: Um einen Snapshot in eine andere Region AMI zu kopieren und den Backing-Snapshot zu verschlüsseln

Der folgende `copy-image` Befehl kopiert den angegebenen Wert AMI aus der `us-west-2` Region in die aktuelle Region und verschlüsselt den Backing-Snapshot mit dem angegebenen KMS Schlüssel.

```
aws ec2 copy-image \  
  --source-region us-west-2 \  
  --name ami-name \  
  --source-image-id ami-066877671789bd71b \  
  --encrypted \  
  --kms-key-id alias/my-kms-key
```

Ausgabe:

```
{
```

```
"ImageId": "ami-0123456789abcdefg"
}
```

Weitere Informationen finden Sie unter [Kopieren einer AMI](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 3: Um Ihre benutzerdefinierten AMI Tags beim Kopieren eines einzubeziehen AMI

Der folgende `copy-image` Befehl verwendet den `--copy-image-tags` Parameter, um Ihre benutzerdefinierten AMI Tags beim Kopieren von zu kopieren. AMI

```
aws ec2 copy-image \
  --region us-east-1 \
  --name ami-name \
  --source-region us-west-2 \
  --source-image-id ami-066877671789bd71b \
  --description "This is my copied image."
  --copy-image-tags
```

Ausgabe:

```
{
  "ImageId": "ami-0123456789abcdefg"
}
```

Weitere Informationen finden Sie unter [Kopieren einer AMI](#) im EC2Amazon-Benutzerhandbuch.

- APIEinzelheiten finden Sie [CopyImage](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der AMI in der Region „EU (Irland)“ angegebene Wert in die Region „USA West (Oregon)“ kopiert. Wenn `-Region` nicht angegeben ist, wird die aktuelle Standardregion als Zielregion verwendet.

```
Copy-EC2Image -SourceRegion eu-west-1 -SourceImageId ami-12345678 -Region us-
west-2 -Name "Copy of ami-12345678"
```

Ausgabe:

```
ami-87654321
```

- API-Einheiten finden Sie unter [CopyImage AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **CopySnapshot** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung von `CopySnapshot`.

CLI

AWS CLI

Beispiel 1: Um einen Snapshot in eine andere Region zu kopieren

Der folgende `copy-snapshot` Beispielbefehl kopiert den angegebenen Snapshot von der `us-west-2` Region in die `us-east-1` Region und fügt eine kurze Beschreibung hinzu.

```
aws ec2 copy-snapshot \  
  --region us-east-1 \  
  --source-region us-west-2 \  
  --source-snapshot-id snap-066877671789bd71b \  
  --description "This is my copied snapshot."
```

Ausgabe:

```
{  
  "SnapshotId": "snap-066877671789bd71b"  
}
```

Weitere Informationen finden Sie unter [Kopieren eines EBS Amazon-Snapshots](#) im EC2 Amazon-Benutzerhandbuch.

Beispiel 2: Um einen unverschlüsselten Snapshot zu kopieren und den neuen Snapshot zu verschlüsseln

Der folgende `copy-snapshot` Befehl kopiert den angegebenen unverschlüsselten Snapshot aus der `us-west-2` Region in die aktuelle Region und verschlüsselt den neuen Snapshot mit dem angegebenen Schlüssel. KMS

```
aws ec2 copy-snapshot \  
  --source-region us-west-2 \  
  --source-snapshot-id snap-066877671789bd71b \  
  --encrypted \  
  --kms-key-id alias/my-kms-key
```

Ausgabe:

```
{  
  "SnapshotId": "snap-066877671789bd71b"  
}
```

Weitere Informationen finden Sie unter [Kopieren eines EBS Amazon-Snapshots](#) im EC2Amazon-Benutzerhandbuch.

- APIEinzelheiten finden Sie [CopySnapshot](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Snapshot aus der Region EU (Irland) in die Region USA West (Oregon) kopiert.

```
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678 -Region  
us-west-2
```

Beispiel 2: Wenn Sie eine Standardregion festlegen und den Parameter `Region` weglassen, ist die Standardzielregion die Standardregion.

```
Set-DefaultAWSRegion us-west-2  
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678
```

- APIEinzelheiten finden Sie unter [CopySnapshot AWS Tools for PowerShell Cmdlet](#)-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter. [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **CreateCapacityReservation** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CreateCapacityReservation`.

CLI

AWS CLI

Beispiel 1: Um eine Kapazitätsreservierung zu erstellen

Im folgenden `create-capacity-reservation` Beispiel wird eine Kapazitätsreservierung in der `eu-west-1a` Availability Zone erstellt, in der Sie drei `t2.medium` Instances starten können, auf denen ein Linux/Unix-Betriebssystem ausgeführt wird. Standardmäßig wird die Kapazitätsreservierung mit offenen Instance-Übereinstimmungskriterien und ohne Unterstützung für kurzlebigen Speicher erstellt. Sie bleibt aktiv, bis Sie sie manuell stornieren.

```
aws ec2 create-capacity-reservation \
  --availability-zone eu-west-1a \
  --instance-type t2.medium \
  --instance-platform Linux/UNIX \
  --instance-count 3
```

Ausgabe:

```
{
  "CapacityReservation": {
    "CapacityReservationId": "cr-1234abcd56EXAMPLE ",
    "EndDateType": "unlimited",
    "AvailabilityZone": "eu-west-1a",
    "InstanceMatchCriteria": "open",
    "EphemeralStorage": false,
    "CreateDate": "2019-08-16T09:27:35.000Z",
    "AvailableInstanceCount": 3,
    "InstancePlatform": "Linux/UNIX",
    "TotalInstanceCount": 3,
    "State": "active",
    "Tenancy": "default",
```

```
    "EbsOptimized": false,  
    "InstanceType": "t2.medium"  
  }  
}
```

Beispiel 2: Um eine Kapazitätsreservierung zu erstellen, die automatisch an einem bestimmten Datum/einer bestimmten Uhrzeit endet

Im folgenden `create-capacity-reservation` Beispiel wird eine Kapazitätsreservierung in der `eu-west-1a` Availability Zone erstellt, in der Sie drei `m5.large` Instances starten können, auf denen ein Linux/Unix-Betriebssystem ausgeführt wird. Diese Kapazitätsreservierung endet automatisch am 31.08.2019 um 23:59:59 Uhr.

```
aws ec2 create-capacity-reservation \  
  --availability-zone eu-west-1a \  
  --instance-type m5.large \  
  --instance-platform Linux/UNIX \  
  --instance-count 3 \  
  --end-date-type limited \  
  --end-date 2019-08-31T23:59:59Z
```

Ausgabe:

```
{  
  "CapacityReservation": {  
    "CapacityReservationId": "cr-1234abcd56EXAMPLE ",  
    "EndDateType": "limited",  
    "AvailabilityZone": "eu-west-1a",  
    "EndDate": "2019-08-31T23:59:59.000Z",  
    "InstanceMatchCriteria": "open",  
    "EphemeralStorage": false,  
    "CreateDate": "2019-08-16T10:15:53.000Z",  
    "AvailableInstanceCount": 3,  
    "InstancePlatform": "Linux/UNIX",  
    "TotalInstanceCount": 3,  
    "State": "active",  
    "Tenancy": "default",  
    "EbsOptimized": false,  
    "InstanceType": "m5.large"  
  }  
}
```

Beispiel 3: So erstellen Sie eine Kapazitätsreservierung, die nur gezielte Instance-Starts akzeptiert

Im folgenden `create-capacity-reservation` Beispiel wird eine Kapazitätsreservierung erstellt, die nur gezielte Instance-Starts akzeptiert.

```
aws ec2 create-capacity-reservation \  
  --availability-zone eu-west-1a \  
  --instance-type m5.large \  
  --instance-platform Linux/UNIX \  
  --instance-count 3 \  
  --instance-match-criteria targeted
```

Ausgabe:

```
{  
  "CapacityReservation": {  
    "CapacityReservationId": "cr-1234abcd56EXAMPLE ",  
    "EndDateType": "unlimited",  
    "AvailabilityZone": "eu-west-1a",  
    "InstanceMatchCriteria": "targeted",  
    "EphemeralStorage": false,  
    "CreateDate": "2019-08-16T10:21:57.000Z",  
    "AvailableInstanceCount": 3,  
    "InstancePlatform": "Linux/UNIX",  
    "TotalInstanceCount": 3,  
    "State": "active",  
    "Tenancy": "default",  
    "EbsOptimized": false,  
    "InstanceType": "m5.large"  
  }  
}
```

Weitere Informationen finden Sie unter [Erstellen einer Kapazitätsreservierung](#) im Amazon Elastic Compute Cloud-Benutzerhandbuch für Linux-Instances.

- API-Einheiten finden Sie [CreateCapacityReservation](#) unter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine neue Kapazitätsreservierung mit den angegebenen Attributen erstellt

```
Add-EC2CapacityReservation -InstanceType m4.xlarge -InstanceCount 2 -
AvailabilityZone eu-west-1b -EbsOptimized True -InstancePlatform Windows
```

Ausgabe:

```
AvailabilityZone      : eu-west-1b
AvailableInstanceCount : 2
CapacityReservationId : cr-0c1f2345db6f7cdba
CreateDate            : 3/28/2019 9:29:41 AM
EbsOptimized         : True
EndDate               : 1/1/0001 12:00:00 AM
EndDateType          : unlimited
EphemeralStorage     : False
InstanceMatchCriteria : open
InstancePlatform     : Windows
InstanceType         : m4.xlarge
State                 : active
Tags                  : {}
Tenancy               : default
TotalInstanceCount   : 2
```

- API-Einheiten finden Sie unter [CreateCapacityReservation AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **CreateCustomerGateway** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CreateCustomerGateway`.

CLI

AWS CLI

Um ein Kunden-Gateway zu erstellen

In diesem Beispiel wird ein Kunden-Gateway mit der angegebenen IP-Adresse für die externe Schnittstelle erstellt.

Befehl:

```
aws ec2 create-customer-gateway --type ipsec.1 --public-ip 12.1.2.3 --bgp-  
asn 65534
```

Ausgabe:

```
{  
  "CustomerGateway": {  
    "CustomerGatewayId": "cgw-0e11f167",  
    "IpAddress": "12.1.2.3",  
    "State": "available",  
    "Type": "ipsec.1",  
    "BgpAsn": "65534"  
  }  
}
```

- API-Einheiten finden Sie [CreateCustomerGateway](#) unter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Kunden-Gateway erstellt.

```
New-EC2CustomerGateway -Type ipsec.1 -PublicIp 203.0.113.12 -BgpAsn 65534
```

Ausgabe:

```
BgpAsn           : 65534  
CustomerGatewayId : cgw-1a2b3c4d  
IpAddress        : 203.0.113.12
```

```
State      : available
Tags       : {}
Type       : ipsec.1
```

- API-Einheiten finden Sie unter [CreateCustomerGateway AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **CreateDhcpOptions** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CreateDhcpOptions`.

CLI

AWS CLI

Um eine Reihe von DHCP-Optionen zu erstellen

Im folgenden `create-dhcp-options` Beispiel wird eine Reihe von DHCP-Optionen erstellt, die den Domännennamen, die Domännennamenserver und den BIOS-Netzwerktyp angeben.

```
aws ec2 create-dhcp-options \
  --dhcp-configuration \
    "Key=domain-name-servers,Values=10.2.5.1,10.2.5.2" \
    "Key=domain-name,Values=example.com" \
    "Key=netbios-node-type,Values=2"
```

Ausgabe:

```
{
  "DhcpOptions": {
    "DhcpConfigurations": [
      {
        "Key": "domain-name",
        "Values": [
          {
            "Value": "example.com"
          }
        ]
      }
    ]
  }
}
```

```

    ]
  },
  {
    "Key": "domain-name-servers",
    "Values": [
      {
        "Value": "10.2.5.1"
      },
      {
        "Value": "10.2.5.2"
      }
    ]
  },
  {
    "Key": "netbios-node-type",
    "Values": [
      {
        "Value": "2"
      }
    ]
  }
],
"DhcpOptionsId": "dopt-06d52773eff4c55f3"
}
}

```

- API-Einheiten finden Sie [CreateDhcpOptions](#) unter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene DHCP Optionssatz erstellt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```

$options = @( @{Key="domain-name";Values=@("abc.local")}, @{Key="domain-name-servers";Values=@("10.0.0.101","10.0.0.102")} )
New-EC2DhcpOption -DhcpConfiguration $options

```

Ausgabe:

DhcpConfigurations	DhcpOptionsId	Tags
--------------------	---------------	------

```

-----
{domain-name, domain-name-servers}    dopt-1a2b3c4d    {}

```

Beispiel 2: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um jede DHCP Option zu erstellen.

```

$option1 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option1.Key = "domain-name"
$option1.Values = "abc.local"

$option2 = New-Object Amazon.EC2.Model.DhcpConfiguration
$option2.Key = "domain-name-servers"
$option2.Values = @"10.0.0.101", "10.0.0.102"@

New-EC2DhcpOption -DhcpConfiguration @($option1, $option2)

```

Ausgabe:

```

DhcpConfigurations                DhcpOptionsId    Tags
-----
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}

```

- API-Einheiten finden Sie unter [CreateDhcpOptions AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **CreateFlowLogs** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CreateFlowLogs`.

CLI

AWS CLI

Beispiel 1: Um ein Flow-Protokoll zu erstellen

Im folgenden `create-flow-logs` Beispiel wird ein Flow-Protokoll erstellt, das den gesamten zurückgewiesenen Datenverkehr für die angegebene Netzwerkschnittstelle erfasst. Die

Flow-Protokolle werden mithilfe der Berechtigungen in der angegebenen IAM Rolle an eine Protokollgruppe in CloudWatch Logs übermittelt.

```
aws ec2 create-flow-logs \
  --resource-type NetworkInterface \
  --resource-ids eni-11223344556677889 \
  --traffic-type REJECT \
  --log-group-name my-flow-logs \
  --deliver-logs-permission-arn arn:aws:iam::123456789101:role/publishFlowLogs
```

Ausgabe:

```
{
  "ClientToken": "so0eNA2uSHUNlHI0S2cJ305GuIX1CezaRdGtexample",
  "FlowLogIds": [
    "fl-12345678901234567"
  ],
  "Unsuccessful": []
}
```

Weitere Informationen finden Sie unter [VPCFlow Logs](#) im VPCAmazon-Benutzerhandbuch.

Beispiel 2: Um ein Flow-Protokoll mit einem benutzerdefinierten Format zu erstellen

Das folgende `create-flow-logs` Beispiel erstellt ein Flow-Protokoll, das den gesamten Datenverkehr für die angegebenen Daten erfasst VPC und die Flow-Logs an einen Amazon S3 S3-Bucket übermittelt. Der Parameter `--log-format` legt ein benutzerdefiniertes Format für die Flow-Protokoll Datensätze fest. Um diesen Befehl unter Windows auszuführen, ändern Sie die einfachen Anführungszeichen (') in doppelte Anführungszeichen (").

```
aws ec2 create-flow-logs \
  --resource-type VPC \
  --resource-ids vpc-00112233344556677 \
  --traffic-type ALL \
  --log-destination-type s3 \
  --log-destination arn:aws:s3:::flow-log-bucket/my-custom-flow-logs/ \
  --log-format '${version} ${vpc-id} ${subnet-id} ${instance-id} ${srcaddr}
  ${dstaddr} ${srcport} ${dstport} ${protocol} ${tcp-flags} ${type} ${pkt-srcaddr}
  ${pkt-dstaddr}'
```

Weitere Informationen finden Sie unter [VPCFlow Logs](#) im VPCAmazon-Benutzerhandbuch.

Beispiel 3: Um ein Flow-Protokoll mit einem maximalen Aggregationsintervall von einer Minute zu erstellen

Das folgende `create-flow-logs` Beispiel erstellt ein Flow-Protokoll, das den gesamten Datenverkehr für die angegebenen Daten erfasst VPC und die Flow-Logs an einen Amazon S3 S3-Bucket übermittelt. Der `--max-aggregation-interval` Parameter gibt ein maximales Aggregationsintervall von 60 Sekunden (1 Minute) an.

```
aws ec2 create-flow-logs \
  --resource-type VPC \
  --resource-ids vpc-00112233344556677 \
  --traffic-type ALL \
  --log-destination-type s3 \
  --log-destination arn:aws:s3:::flow-log-bucket/my-custom-flow-logs/ \
  --max-aggregation-interval 60
```

Weitere Informationen finden Sie unter [VPCFlow Logs](#) im VPCAmazon-Benutzerhandbuch.

- API-Einheiten finden Sie [CreateFlowLogs](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird mithilfe der Berechtigungen der Rolle „Admin“ ein EC2 Flowlog für das Subnetz Subnetz-1d234567 bis zum cloud-watch-log Namen „subnet1-log“ für den gesamten Datenverkehr erstellt REJECT

```
New-EC2FlowLog -ResourceId "subnet-1d234567" -LogDestinationType cloud-watch-logs -LogGroupName subnet1-log -TrafficType "REJECT" -ResourceType Subnet -DeliverLogsPermissionArn "arn:aws:iam::98765432109:role/Admin"
```

Ausgabe:

```
ClientToken                               FlowLogIds                                Unsuccessful
-----
m1VN2cxP3iB4qo//VUK15EU6cF7gQL0xcqNefvjeTGw= {f1-012fc34eed5678c9d} {}
```

- [CreateFlowLogs AWS Tools for PowerShell](#) Einzelheiten finden Sie unter Cmdlet-Referenz.
API

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **CreateImage** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CreateImage`.

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine AMI von einer EBS Amazon-gestützten Instance

Im folgenden `create-image` Beispiel wird AMI aus der angegebenen Instance eine erstellt.

```
aws ec2 create-image \  
  --instance-id i-1234567890abcdef0 \  
  --name "My server" \  
  --description "An AMI for my server"
```

Ausgabe:

```
{  
  "ImageId": "ami-abcdef01234567890"  
}
```

Weitere Informationen zur Angabe einer Blockgerätezuweisung für Sie AMI finden Sie unter [Angaben einer Blockgerätezuweisung für eine AMI](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 2: So erstellen Sie eine AMI aus einer EBS Amazon-gestützten Instance ohne Neustart

Das folgende `create-image` Beispiel erstellt eine AMI und legt den Parameter `--no-reboot` fest, sodass die Instance nicht neu gestartet wird, bevor das Image erstellt wurde.

```
aws ec2 create-image \  
  --instance-id i-1234567890abcdef0 \  
  --name "My server" \  
  --no-reboot
```

Ausgabe:

```
{
  "ImageId": "ami-abcdef01234567890"
}
```

Weitere Informationen zur Angabe einer Blockgerätezuweisung für Sie AMI finden Sie unter [Angeben einer Blockgerätezuweisung für eine AMI](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 3: Um ein AMI und Schnappschüsse bei der Erstellung zu taggen

Im folgenden `create-image` Beispiel wird einAMI, AMI und die Schnappschüsse werden mit demselben Tag versehen `cost-center=cc123`

```
aws ec2 create-image \
  --instance-id i-1234567890abcdef0 \
  --name "My server" \
  --tag-specifications "ResourceType=image,Tags=[{Key=cost-center,Value=cc123}]"" "ResourceType=snapshot,Tags=[{Key=cost-center,Value=cc123}]""
```

Ausgabe:

```
{
  "ImageId": "ami-abcdef01234567890"
}
```

Weitere Informationen zum Taggen Ihrer Ressourcen bei der Erstellung finden [Sie unter Hinzufügen von Tags bei der Erstellung von Ressourcen](#) im EC2Amazon-Benutzerhandbuch.

- APIEinzelheiten finden Sie [CreateImage](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird aus der angegebenen Instanz eine AMI mit dem angegebenen Namen und der angegebenen Beschreibung erstellt. Amazon EC2 versucht, die Instance sauber herunterzufahren, bevor das Image erstellt wird, und startet die Instance nach Abschluss neu.


```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
server AMI"
```

Beispiel 2: In diesem Beispiel wird aus der angegebenen Instance eine AMI mit dem angegebenen Namen und der Beschreibung erstellt. Amazon EC2 erstellt das Image, ohne die Instance herunterzufahren und neu zu starten. Daher kann die Dateisystemintegrität des erstellten Images nicht garantiert werden.

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
server AMI" -NoReboot $true
```

Beispiel 3: In diesem Beispiel wird ein AMI mit drei Volumes erstellt. Das erste Volume basiert auf einem EBS Amazon-Snapshot. Das zweite Volume ist ein leeres EBS 100-GiB-Amazon-Volume. Das dritte Volume ist ein Instance-Speicher-Volume. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ebsBlock1 = @{SnapshotId="snap-1a2b3c4d"}
$ebsBlock2 = @{VolumeSize=100}

New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description
  "My web server AMI" -BlockDeviceMapping @( @{DeviceName="/dev/sdf";Ebs=
  $ebsBlock1}, @{DeviceName="/dev/sdg";Ebs=$ebsBlock2}, @{DeviceName="/dev/
  sdc";VirtualName="ephemeral0"})
```

- API-Einheiten finden Sie unter [CreateImage AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **CreateInstanceExportTask** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CreateInstanceExportTask`.

CLI

AWS CLI

Um eine Instanz zu exportieren

Dieser Beispielbefehl erstellt eine Aufgabe zum Exportieren der Instance `i-1234567890abcdef0` in den Amazon S3 S3-Bucket `myexportbucket`.

Befehl:

```
aws ec2 create-instance-export-task --description "RHEL5 instance" --instance-id i-1234567890abcdef0 --target-environment vmware --export-to-s3-task DiskImageFormat=vmdk,ContainerFormat=ova,S3Bucket=myexportbucket,S3Prefix=RHEL5
```

Ausgabe:

```
{
  "ExportTask": {
    "State": "active",
    "InstanceExportDetails": {
      "InstanceId": "i-1234567890abcdef0",
      "TargetEnvironment": "vmware"
    },
    "ExportToS3Task": {
      "S3Bucket": "myexportbucket",
      "S3Key": "RHEL5export-i-fh8sjjsq.ova",
      "DiskImageFormat": "vmdk",
      "ContainerFormat": "ova"
    },
    "Description": "RHEL5 instance",
    "ExportTaskId": "export-i-fh8sjjsq"
  }
}
```

- API-Einheiten finden Sie unter Befehlsreferenz. [CreateInstanceExportTask](#) AWS CLI

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine `i-0800b00a00EXAMPLE` gestoppte Instanz als virtuelle Festplatte (VHD) in den S3-Bucket `testbucket-export-instances-2019` exportiert. Die Zielumgebung ist `Microsoft`, und der Regionsparameter wird hinzugefügt, weil sich die Instanz in der `us-east-1` Region befindet, während die AWS Standardregion des Benutzers nicht `us-east-1` ist. Um den Status der Exportaufgabe abzurufen, kopieren Sie den `ExportTaskId` Wert aus den Ergebnissen dieses

Befehls und führen Sie dann den Befehl aus **Get-EC2ExportTask -ExportTaskId export_task_ID_from_results**.

```
New-EC2InstanceExportTask -InstanceId i-0800b00a00EXAMPLE -
ExportToS3Task_DiskImageFormat VHD -ExportToS3Task_S3Bucket "amzn-s3-demo-bucket"
-TargetEnvironment Microsoft -Region us-east-1
```

Ausgabe:

```
Description          :
ExportTaskId         : export-i-077c73108aEXAMPLE
ExportToS3Task       : Amazon.EC2.Model.ExportToS3Task
InstanceExportDetails : Amazon.EC2.Model.InstanceExportDetails
State                : active
StatusMessage        :
```

- APIEinzelheiten finden Sie unter [CreateInstanceExportTask AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **CreateInternetGateway** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CreateInternetGateway`.

CLI

AWS CLI

Um ein Internet-Gateway zu erstellen

Im folgenden `create-internet-gateway` Beispiel wird ein Internet-Gateway mit dem Tag `erstelltName=my-igw` erstellt.

```
aws ec2 create-internet-gateway \
  --tag-specifications ResourceType=internet-gateway, Tags=[{Key=Name, Value=my-igw}]
```

Ausgabe:

```
{
  "InternetGateway": {
    "Attachments": [],
    "InternetGatewayId": "igw-0d0fb496b3994d755",
    "OwnerId": "123456789012",
    "Tags": [
      {
        "Key": "Name",
        "Value": "my-igw"
      }
    ]
  }
}
```

Weitere Informationen finden Sie unter [Internet-Gateways](#) im VPCAmazon-Benutzerhandbuch.

- APIEinzelheiten finden Sie [CreateInternetGateway](#) in der AWS CLI Befehlsreferenz.

PowerShell**Tools für PowerShell**

Beispiel 1: In diesem Beispiel wird ein Internet-Gateway erstellt.

```
New-EC2InternetGateway
```

Ausgabe:

Attachments	InternetGatewayId	Tags
{}	igw-1a2b3c4d	{}

- APIEinzelheiten finden Sie unter [CreateInternetGateway AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter. [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie es **CreateKeyPair** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `CreateKeyPair`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Create an Amazon EC2 key pair with a specified name.
/// </summary>
/// <param name="keyPairName">The name for the new key pair.</param>
/// <returns>The Amazon EC2 key pair created.</returns>
public async Task<KeyPair?> CreateKeyPair(string keyPairName)
{
    try
    {
        var request = new CreateKeyPairRequest { KeyName = keyPairName, };

        var response = await _amazonEC2.CreateKeyPairAsync(request);

        var kp = response.KeyPair;
        // Return the key pair so it can be saved if needed.

        // Wait until the key pair exists.
        int retries = 5;
        while (retries-- > 0)
        {
            Console.WriteLine($"Checking for new KeyPair {keyPairName}...");
            var keyPairs = await DescribeKeyPairs(keyPairName);
            if (keyPairs.Any())
```

```
        {
            return kp;
        }

        Thread.Sleep(5000);
        retries--;
    }
    _logger.LogError($"Unable to find newly created KeyPair
{keyPairName}.");
    throw new DoesNotExistException("KeyPair not found");
}
catch (AmazonEC2Exception ec2Exception)
{
    if (ec2Exception.ErrorCode == "InvalidKeyPair.Duplicate")
    {
        _logger.LogError(
            $"A key pair called {keyPairName} already exists.");
    }

    throw;
}
catch (Exception ex)
{
    _logger.LogError(
        $"An error occurred while creating the key pair.: {ex.Message}");
    throw;
}
}

/// <summary>
/// Save KeyPair information to a temporary file.
/// </summary>
/// <param name="keyPair">The name of the key pair.</param>
/// <returns>The full path to the temporary file.</returns>
public string SaveKeyPair(KeyPair keyPair)
{
    var tempPath = Path.GetTempPath();
    var tempFileName = $"{tempPath}\\{Path.GetRandomFileName()}";
    var pemFileName = Path.ChangeExtension(tempFileName, "pem");

    // Save the key pair to a file in a temporary folder.
    using var stream = new FileStream(pemFileName, FileMode.Create);
    using var writer = new StreamWriter(stream);
    writer.WriteLine(keyPair.KeyMaterial);
}
```

```

    return pemFileName;
}

```

- API-Einheiten finden Sie [CreateKeyPair](#) unter AWS SDK for .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

#####
# function ec2_create_keypair
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or
# 2048-bit RSA key pair
# and writes it to a file.
#
# Parameters:
#     -n key_pair_name - A key pair name.
#     -f file_path - File to store the key pair.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_create_keypair() {
    local key_pair_name file_path response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_create_keypair"
        echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) ED25519 or 2048-
bit RSA key pair"
        echo " and writes it to a file."
    }
}

```

```
    echo " -n key_pair_name - A key pair name."
    echo " -f file_path - File to store the key pair."
    echo ""
}

# Retrieve the calling parameters.
while getopts "n:f:h" option; do
    case "${option}" in
        n) key_pair_name="${OPTARG}" ;;
        f) file_path="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

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

if [[ -z "$file_path" ]]; then
    errecho "ERROR: You must provide a file path with the -f parameter."
    usage
    return 1
fi

response=$(aws ec2 create-key-pair \
    --key-name "$key_pair_name" \
    --query 'KeyMaterial' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
}
```



```

if [[ -n "$file_path" ]]; then
    echo "$response" >"$file_path"
fi

return 0
}

```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then

```

```

    errecho " The system environment or configuration was invalid."
elif [ "$err_code" == 254 ]; then
    errecho " The service returned an error."
elif [ "$err_code" == 255 ]; then
    errecho " 255 is a catch-all error."
fi

return 0
}

```

- API-Einheiten finden Sie [CreateKeyPair](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

//! Create an Amazon Elastic Compute Cloud (Amazon EC2) instance key pair.
/*!
  \param keyPairName: A name for a key pair.
  \param keyFilePath: File path where the credentials are stored. Ignored if it
  is an empty string;
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::EC2::createKeyPair(const Aws::String &keyPairName, const Aws::String
&keyFilePath,
                               const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::CreateKeyPairRequest request;
    request.SetKeyName(keyPairName);

    Aws::EC2::Model::CreateKeyPairOutcome outcome =
    ec2Client.CreateKeyPair(request);
    if (!outcome.IsSuccess()) {

```

```
        std::cerr << "Failed to create key pair - " << keyPairName << ". " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully created key pair named " <<
            keyPairName << std::endl;
        if (!keyFilePath.empty()) {
            std::ofstream keyFile(keyFilePath.c_str());
            keyFile << outcome.GetResult().GetKeyMaterial();
            keyFile.close();
            std::cout << "Keys written to the file " <<
                keyFilePath << std::endl;
        }
    }

    return outcome.IsSuccess();
}
```

- API-Einheiten finden Sie [CreateKeyPair](#) unter AWS SDK for C++ API-Referenz.

CLI

AWS CLI

So erstellen Sie ein Schlüsselpaar

In diesem Beispiel wird eine Schlüsselrolle mit dem Namen `MyKeyPair` erstellt.

Befehl:

```
aws ec2 create-key-pair --key-name MyKeyPair
```


Die Ausgabe ist eine ASCII-Version des privaten Schlüssels und des Schlüsselfingerabdrucks. Sie müssen den Schlüssel in einer Datei speichern.

Weitere Informationen finden Sie unter Verwenden von Schlüsselpaaren im Benutzerhandbuch für die AWS-Befehlszeilenschnittstelle.

- API-Einheiten finden Sie [CreateKeyPair](#) in der AWS CLI-Befehlsreferenz.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Creates a new key pair asynchronously.
 *
 * @param keyName the name of the key pair to create
 * @param fileName the name of the file to write the key material to
 * @return a {@link CompletableFuture} that represents the asynchronous
operation
 *         of creating the key pair and writing the key material to a file
 */
public CompletableFuture<CreateKeyPairResponse> createKeyPairAsync(String
keyName, String fileName) {
    CreateKeyPairRequest request = CreateKeyPairRequest.builder()
        .keyName(keyName)
        .build();

    CompletableFuture<CreateKeyPairResponse> responseFuture =
getAsyncClient().createKeyPair(request);
    responseFuture.whenComplete((response, exception) -> {
        if (response != null) {
            try {
                BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName));
                writer.write(response.keyMaterial());
                writer.close();
            } catch (IOException e) {
                throw new RuntimeException("Failed to write key material to
file: " + e.getMessage(), e);
            }
        } else {
            throw new RuntimeException("Failed to create key pair: " +
exception.getMessage(), exception);
        }
    });
}
```

```
    });  
  
    return responseFuture;  
}
```

- API-Einheiten finden Sie [CreateKeyPair](#) unter AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import { CreateKeyPairCommand, EC2Client } from "@aws-sdk/client-ec2";  
  
/**  
 * Creates an ED25519 or 2048-bit RSA key pair with the specified name and in the  
 * specified PEM or PPK format.  
 * Amazon EC2 stores the public key and displays the private key for you to save  
 * to a file.  
 * @param {{ keyName: string }} options  
 */  
export const main = async ({ keyName }) => {  
  const client = new EC2Client({});  
  const command = new CreateKeyPairCommand({  
    KeyName: keyName,  
  });  
  
  try {  
    const { KeyMaterial, KeyName } = await client.send(command);  
    console.log(KeyName);  
    console.log(KeyMaterial);  
  } catch (caught) {  
    if (caught instanceof Error && caught.name === "InvalidKeyPair.Duplicate") {  
      console.warn(`${caught.message}. Try another key name.`);  
    } else {  
      throw caught;  
    }  
  }  
}
```

```
    }  
  }  
};
```

- API-Einheiten finden Sie [CreateKeyPair](#) unter AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
suspend fun createEC2KeyPair(keyNameVal: String) {  
    val request =  
        CreateKeyPairRequest {  
            keyName = keyNameVal  
        }  
  
    Ec2Client { region = "us-west-2" }.use { ec2 ->  
        val response = ec2.createKeyPair(request)  
        println("The key ID is ${response.keyPairId}")  
    }  
}
```

- API-Einheiten finden Sie [CreateKeyPair](#) in der AWS SDK API Kotlin-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel erstellt ein key pair und erfasst den PEM -codierten RSA privaten Schlüssel in einer Datei mit dem angegebenen Namen. Wenn Sie verwenden PowerShell, muss die Kodierung auf ASCII eingestellt sein, um einen gültigen Schlüssel zu generieren. Weitere Informationen finden Sie unter EC2 Amazon-Schlüsselpaare (<https://>

docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-keypairs.html) erstellen, anzeigen und löschen im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle.

```
(New-EC2KeyPair -KeyName "my-key-pair").KeyMaterial | Out-File -Encoding ascii -
FilePath C:\path\my-key-pair.pem
```

- API-Einheiten finden Sie unter [CreateKeyPair AWS Tools for PowerShell](#) Cmdlet-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
        directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is
        stored.
                           This should be a secure folder.
```

```
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                       This is a high-level object that wraps key pair actions.
Optional.
    """
    self.ec2_client = ec2_client
    self.key_pair = key_pair
    self.key_file_path: Optional[str] = None
    self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2
client
and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())

    def create(self, key_name: str) -> dict:
        """
        Creates a key pair that can be used to securely connect to an EC2
instance.
        The returned key pair contains private key information that cannot be
retrieved
again. The private key data is stored as a .pem file.

        :param key_name: The name of the key pair to create.
        :return: A dictionary representing the Boto3 KeyPair object that
represents the newly created key pair.
        :raises ClientError: If there is an error in creating the key pair, for
example, if a key pair with the same name already exists.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_name)
            self.key_pair = response
            self.key_file_path = os.path.join(
                self.key_file_dir.name, f"{self.key_pair['KeyName']}.pem"
            )
            with open(self.key_file_path, "w") as key_file:
                key_file.write(self.key_pair["KeyMaterial"])
```



```

except ClientError as err:
    if err.response["Error"]["Code"] == "InvalidKeyPair.Duplicate":
        logger.error(
            f"A key pair called {key_name} already exists. "
            "Please choose a different name for your key pair "
            "or delete the existing key pair before creating."
        )
        raise
    else:
        return self.key_pair

```

- API-Einheiten finden Sie unter [CreateKeyPair](#) Python (Boto3) API-Referenz. AWS SDK

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

# This code example does the following:
# 1. Creates a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
# 2. Displays information about available key pairs.
# 3. Deletes the key pair.

require 'aws-sdk-ec2'

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name for the key pair and private
#   key file.
# @return [Boolean] true if the key pair and private key file were
#   created; otherwise, false.
# @example
#   exit 1 unless key_pair_created?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'

```

```
# )
def key_pair_created?(ec2_client, key_pair_name)
  key_pair = ec2_client.create_key_pair(key_name: key_pair_name)
  puts "Created key pair '#{key_pair.key_name}' with fingerprint " \
    "'#{key_pair.key_fingerprint}' and ID '#{key_pair.key_pair_id}'."
  filename = File.join(Dir.home, "#{key_pair_name}.pem")
  File.open(filename, 'w') { |file| file.write(key_pair.key_material) }
  puts "Private key file saved locally as '#{filename}'."
  true
rescue Aws::EC2::Errors::InvalidKeyPairDuplicate
  puts "Error creating key pair: a key pair named '#{key_pair_name}' " \
    'already exists.'
  false
rescue StandardError => e
  puts "Error creating key pair or saving private key file: #{e.message}"
  false
end

# Displays information about available key pairs in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   describe_key_pairs(Aws::EC2::Client.new(region: 'us-west-2'))
def describe_key_pairs(ec2_client)
  result = ec2_client.describe_key_pairs
  if result.key_pairs.count.zero?
    puts 'No key pairs found.'
  else
    puts 'Key pair names:'
    result.key_pairs.each do |key_pair|
      puts key_pair.key_name
    end
  end
end
rescue StandardError => e
  puts "Error getting information about key pairs: #{e.message}"
end

# Deletes a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - The key pair to delete.
#
```

```
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name of the key pair to delete.
# @return [Boolean] true if the key pair was deleted; otherwise, false.
# @example
#   exit 1 unless key_pair_deleted?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_deleted?(ec2_client, key_pair_name)
  ec2_client.delete_key_pair(key_name: key_pair_name)
  true
rescue StandardError => e
  puts "Error deleting key pair: #{e.message}"
  false
end

# Example usage:
def run_me
  key_pair_name = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-key-pairs.rb KEY_PAIR_NAME REGION'
    puts 'Example: ruby ec2-ruby-example-key-pairs.rb my-key-pair us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    key_pair_name = 'my-key-pair'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    key_pair_name = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts 'Displaying existing key pair names before creating this key pair...'
  describe_key_pairs(ec2_client)

  puts '-' * 10
  puts 'Creating key pair...'
  unless key_pair_created?(ec2_client, key_pair_name)
```

```
    puts 'Stopping program.'
    exit 1
end

puts '-' * 10
puts 'Displaying existing key pair names after creating this key pair...'
describe_key_pairs(ec2_client)

puts '-' * 10
puts 'Deleting key pair...'
unless key_pair_deleted?(ec2_client, key_pair_name)
  puts 'Stopping program. You must delete the key pair yourself.'
  exit 1
end
puts 'Key pair deleted.'

puts '-' * 10
puts 'Now that the key pair is deleted, ' \
     'also deleting the related private key pair file...'
filename = File.join(Dir.home, "#{key_pair_name}.pem")
File.delete(filename)
if File.exist?(filename)
  puts "Could not delete file at '#{filename}'. You must delete it yourself."
else
  puts 'File deleted.'
end

puts '-' * 10
puts 'Displaying existing key pair names after deleting this key pair...'
describe_key_pairs(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- API-Einheiten finden Sie [CreateKeyPair](#) unter AWS SDK for Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Rust-Implementierung, die `create_key_pair` des EC2 Clients aufruft und das zurückgegebene Material extrahiert.

```
pub async fn create_key_pair(&self, name: String) -> Result<(KeyPairInfo,
String), EC2Error> {
    tracing::info!("Creating key pair {name}");
    let output = self.client.create_key_pair().key_name(name).send().await?;
    let info = KeyPairInfo::builder()
        .set_key_name(output.key_name)
        .set_key_fingerprint(output.key_fingerprint)
        .set_key_pair_id(output.key_pair_id)
        .build();
    let material = output
        .key_material
        .ok_or_else(|| EC2Error::new("Create Key Pair has no key
material"))?;
    Ok((info, material))
}
```

Eine Funktion, die das `create_key`-Impl aufruft und den privaten Schlüssel sicher speichert.
PEM

```
/// Creates a key pair that can be used to securely connect to an EC2
instance.
/// The returned key pair contains private key information that cannot be
retrieved
/// again. The private key data is stored as a .pem file.
///
/// :param key_name: The name of the key pair to create.
pub async fn create(
    &mut self,
```

```

    ec2: &EC2,
    util: &Util,
    key_name: String,
) -> Result<KeyPairInfo, EC2Error> {
    let (key_pair, material) = ec2
        .create_key_pair(key_name.clone())
        .await
        .map_err(|e| e.add_message(format!("Couldn't create key
{key_name}")))?;

    let path = self.key_file_dir.join(format!("{key_name}.pem"));

    util.write_secure(&key_name, &path, material)?;

    self.key_file_path = Some(path);
    self.key_pair = key_pair.clone();

    Ok(key_pair)
}

```

- API-Einheiten finden Sie [CreateKeyPair](#) in AWS SDK der Rust-Referenz. API

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

TRY.
    oo_result = lo_ec2->createkeypair( iv_keyname = iv_key_name ).
        " oo_result is returned for testing purposes. "
    MESSAGE 'Amazon EC2 key pair created.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- API-Einheiten finden Sie [CreateKeyPair](#) unter AWS SDKSAPABAPI als Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie es **CreateLaunchTemplate** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `CreateLaunchTemplate`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erstellen und Verwalten eines ausfallsicheren Services](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
/// The launch template specifies a Bash script in its user data field that
runs after
/// the instance is started. This script installs the Python packages and
starts a Python
/// web server on the instance.
/// </summary>
/// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
/// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
```

```
/// <returns>The template object.</returns>
public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
{
    try
    {
        await CreateKeyPair(_keyPairName);
        await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
        _instanceProfileName, instancePolicyPath);

        var startServerText = await File.ReadAllTextAsync(startupScriptPath);
        var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

        var amiLatest = await _amazonSsm.GetParameterAsync(
            new GetParameterRequest() { Name = _amiParam });
        var amiId = amiLatest.Parameter.Value;
        var launchTemplateResponse = await
_amazonEc2.CreateLaunchTemplateAsync(
            new CreateLaunchTemplateRequest()
            {
                LaunchTemplateName = _launchTemplateName,
                LaunchTemplateData = new RequestLaunchTemplateData()
                {
                    InstanceType = _instanceType,
                    ImageId = amiId,
                    IamInstanceProfile =
                        new
LaunchTemplateIamInstanceProfileSpecificationRequest()
                        {
                            Name = _instanceProfileName
                        },
                    KeyName = _keyPairName,
                    UserData = System.Convert.ToBase64String(plainTextBytes)
                }
            });
        return launchTemplateResponse.LaunchTemplate;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
```



```

        {
            _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                $"Please try again with a unique name.");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
        throw;
    }
}

```

- API-Einheiten finden Sie [CreateLaunchTemplate](#) unter AWS SDK for .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So erstellen Sie eine Startvorlage

Im folgenden `create-launch-template` Beispiel wird eine Startvorlage erstellt, die das Subnetz angibt, in dem die Instance gestartet werden soll, der Instance eine öffentliche IP-Adresse und eine IPv6 Adresse zuweist und ein Tag für die Instance erstellt.

```

aws ec2 create-launch-template \
  --launch-template-name TemplateForWebServer \
  --version-description WebVersion1 \
  --launch-template-data '{"NetworkInterfaces":
[{"AssociatePublicIpAddress":true,"DeviceIndex":0,"Ipv6AddressCount":1,"SubnetId":"subnet
[{"ResourceType":"instance","Tags":[{"Key":"purpose","Value":"webserver"}]}]}'

```

Ausgabe:

```

{
  "LaunchTemplate": {
    "LatestVersionNumber": 1,
    "LaunchTemplateId": "lt-01238c059e3466abc",

```

```

    "LaunchTemplateName": "TemplateForWebServer",
    "DefaultVersionNumber": 1,
    "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
    "CreateTime": "2019-01-27T09:13:24.000Z"
  }
}

```

Weitere Informationen finden Sie unter Starten einer Instance von einer Startvorlage im Benutzerhandbuch für Amazon Elastic Compute Cloud. Informationen zum Angeben von Parametern im JSON Format „Quoting“ finden Sie unter Quoting Strings im Command Line Interface User Guide.AWS

Beispiel 2: So erstellen Sie eine Startvorlage für Amazon EC2 Auto Scaling

Im folgenden `create-launch-template` Beispiel wird eine Startvorlage mit mehreren Tags und einer Blockgeräte-Zuordnung erstellt, um beim Start einer Instance ein zusätzliches EBS Volume anzugeben. Geben Sie einen Wert an `Groups`, der den Sicherheitsgruppen für die entspricht, in VPC denen Ihre Auto Scaling Scaling-Gruppe Instances starten wird. Geben Sie die Subnetze VPC und als Eigenschaften der Auto Scaling Scaling-Gruppe an.

```

aws ec2 create-launch-template \
  --launch-template-name TemplateForAutoScaling \
  --version-description AutoScalingVersion1 \
  --launch-template-data '{"NetworkInterfaces":
    [{"DeviceIndex":0,"AssociatePublicIpAddress":true,"Groups":
    ["sg-7c227019,sg-903004f8"],"DeleteOnTermination":true}], "ImageId":"ami-
    b42209de", "InstanceType":"m4.large", "TagSpecifications":
    [{"ResourceType":"instance", "Tags":[{"Key":"environment", "Value":"production"},
    {"Key":"purpose", "Value":"webserver"}]}, {"ResourceType":"volume", "Tags":
    [{"Key":"environment", "Value":"production"}, {"Key":"cost-
    center", "Value":"cc123"}]}], "BlockDeviceMappings":[{"DeviceName":"/dev/
    sda1", "Ebs":{"VolumeSize":100}}]}' --region us-east-1

```

Ausgabe:

```

{
  "LaunchTemplate": {
    "LatestVersionNumber": 1,
    "LaunchTemplateId": "lt-0123c79c33a54e0abc",
    "LaunchTemplateName": "TemplateForAutoScaling",
    "DefaultVersionNumber": 1,
    "CreatedBy": "arn:aws:iam::123456789012:user/Bob",

```

```

    "CreateTime": "2019-04-30T18:16:06.000Z"
  }
}

```

Weitere Informationen finden Sie unter Erstellen einer Startvorlage für eine Auto Scaling Scaling-Gruppe im Amazon EC2 Auto Scaling Scaling-Benutzerhandbuch. Informationen zum Angeben von Parametern im JSON Format „Quoting Strings“ finden Sie unter Quoting Strings im AWS Command Line Interface User Guide.

Beispiel 3: So erstellen Sie eine Startvorlage, die die Verschlüsselung von Volumes spezifiziert EBS

Im folgenden `create-launch-template` Beispiel wird eine Startvorlage erstellt, die verschlüsselte EBS Volumes enthält, die aus einem unverschlüsselten Snapshot erstellt wurden. Außerdem werden die Volumes bei der Erstellung mit Tags versehen. Wenn die standardmäßige Verschlüsselung deaktiviert ist, müssen Sie die `"Encrypted"`-Option wie im folgenden Beispiel gezeigt angeben. Wenn Sie die `"KmsKeyId"` Option verwenden, um ein vom Kunden verwaltetes Objekt anzugeben CMK, müssen Sie die `"Encrypted"` Option auch dann angeben, wenn die Verschlüsselung standardmäßig aktiviert ist.

```

aws ec2 create-launch-template \
  --launch-template-name TemplateForEncryption \
  --launch-template-data file://config.json

```

Inhalt von `config.json`:

```

{
  "BlockDeviceMappings":[
    {
      "DeviceName":"/dev/sda1",
      "Ebs":{"
        "VolumeType":"gp2",
        "DeleteOnTermination":true,
        "SnapshotId":"snap-066877671789bd71b",
        "Encrypted":true,
        "KmsKeyId":"arn:aws:kms:us-east-1:012345678910:key/abcd1234-a123-456a-a12b-a123b4cd56ef"
      }
    }
  ],
  "ImageId":"ami-00068cd7555f543d5",

```

```

    "InstanceType": "c5.large",
    "TagSpecifications": [
      {
        "ResourceType": "volume",
        "Tags": [
          {
            "Key": "encrypted",
            "Value": "yes"
          }
        ]
      }
    ]
  }
}

```

Ausgabe:

```

{
  "LaunchTemplate": {
    "LatestVersionNumber": 1,
    "LaunchTemplateId": "lt-0d5bd51bcf8530abc",
    "LaunchTemplateName": "TemplateForEncryption",
    "DefaultVersionNumber": 1,
    "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
    "CreateTime": "2020-01-07T19:08:36.000Z"
  }
}

```

Weitere Informationen finden Sie unter [Wiederherstellung eines EBS Amazon-Volumes aus einem Snapshot und Standardverschlüsselung im Amazon Elastic Compute Cloud-Benutzerhandbuch](#).

- API-Einheiten finden Sie [CreateLaunchTemplate](#) in der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
const ssmClient = new SSMClient({});
const { Parameter } = await ssmClient.send(
  new GetParameterCommand({
    Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
  }),
);
const ec2Client = new EC2Client({});
await ec2Client.send(
  new CreateLaunchTemplateCommand({
    LaunchTemplateName: NAMES.launchTemplateName,
    LaunchTemplateData: {
      InstanceType: "t3.micro",
      ImageId: Parameter.Value,
      IamInstanceProfile: { Name: NAMES.instanceProfileName },
      UserData: readFileSync(
        join(RESOURCES_PATH, "server_startup_script.sh"),
      ).toString("base64"),
      KeyName: NAMES.keyPairName,
    },
  }),
);
```

- API-Einheiten finden Sie [CreateLaunchTemplate](#) unter AWS SDK for JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

In diesem Beispiel wird eine Startvorlage erstellt, die ein Instance-Profil enthält, das der Instance bestimmte Berechtigungen gewährt, und ein Benutzerdaten-Bash-Skript, das nach dem Start auf der Instance ausgeführt wird.

```
class AutoScalingWrapper:
    """
```

```
Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
"""

def __init__(
    self,
    resource_prefix: str,
    inst_type: str,
    ami_param: str,
    autoscaling_client: boto3.client,
    ec2_client: boto3.client,
    ssm_client: boto3.client,
    iam_client: boto3.client,
):
    """
    Initializes the AutoScaler class with the necessary parameters.

    :param resource_prefix: The prefix for naming AWS resources that are
    created by this class.
    :param inst_type: The type of EC2 instance to create, such as t3.micro.
    :param ami_param: The Systems Manager parameter used to look up the AMI
    that is created.
    :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
    :param ec2_client: A Boto3 EC2 client.
    :param ssm_client: A Boto3 Systems Manager client.
    :param iam_client: A Boto3 IAM client.
    """
    self.inst_type = inst_type
    self.ami_param = ami_param
    self.autoscaling_client = autoscaling_client
    self.ec2_client = ec2_client
    self.ssm_client = ssm_client
    self.iam_client = iam_client
    sts_client = boto3.client("sts")
    self.account_id = sts_client.get_caller_identity()["Account"]

    self.key_pair_name = f"{resource_prefix}-key-pair"
    self.launch_template_name = f"{resource_prefix}-template-"
    self.group_name = f"{resource_prefix}-group"

    # Happy path
    self.instance_policy_name = f"{resource_prefix}-pol"
    self.instance_role_name = f"{resource_prefix}-role"
    self.instance_profile_name = f"{resource_prefix}-prof"
```

```
# Failure mode
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

def create_template(
    self, server_startup_script_file: str, instance_policy_file: str
) -> Dict[str, Any]:
    """
    Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
    Scaling. The
    launch template specifies a Bash script in its user data field that runs
    after
    the instance is started. This script installs Python packages and starts
    a
    Python web server on the instance.

    :param server_startup_script_file: The path to a Bash script file that is
    run
                                     when an instance starts.
    :param instance_policy_file: The path to a file that defines a
    permissions policy
                                 to create and attach to the instance
    profile.
    :return: Information about the newly created template.
    """
    template = {}
    try:
        # Create key pair and instance profile
        self.create_key_pair(self.key_pair_name)
        self.create_instance_profile(
            instance_policy_file,
            self.instance_policy_name,
            self.instance_role_name,
            self.instance_profile_name,
        )

        # Read the startup script
        with open(server_startup_script_file) as file:
            start_server_script = file.read()

        # Get the latest AMI ID
        ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
```

```

ami_id = ami_latest["Parameter"]["Value"]

# Create the launch template
lt_response = self.ec2_client.create_launch_template(
    LaunchTemplateName=self.launch_template_name,
    LaunchTemplateData={
        "InstanceType": self.inst_type,
        "ImageId": ami_id,
        "IamInstanceProfile": {"Name": self.instance_profile_name},
        "UserData": base64.b64encode(
            start_server_script.encode(encoding="utf-8")
        ).decode(encoding="utf-8"),
        "KeyName": self.key_pair_name,
    },
)
template = lt_response["LaunchTemplate"]
log.info(
    f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
)
except ClientError as err:
    log.error(f"Failed to create launch template
{self.launch_template_name}.")
    error_code = err.response["Error"]["Code"]
    if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
        log.info(
            f"Launch template {self.launch_template_name} already exists,
nothing to do."
        )
    log.error(f"Full error:\n\t{err}")
return template

```

- API-Einheiten finden Sie unter [CreateLaunchTemplate](#) Python (Boto3) API -Referenz. AWS SDK

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter. [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie `CreateNetworkAcl` mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CreateNetworkAcl`.

CLI

AWS CLI

Um ein Netzwerk zu erstellen ACL

In diesem Beispiel wird ein Netzwerk ACL für das angegebene Objekt erstelltVPC.

Befehl:

```
aws ec2 create-network-acl --vpc-id vpc-a01106c2
```

Ausgabe:

```
{
  "NetworkAcl": {
    "Associations": [],
    "NetworkAclId": "acl-5fb85d36",
    "VpcId": "vpc-a01106c2",
    "Tags": [],
    "Entries": [
      {
        "CidrBlock": "0.0.0.0/0",
        "RuleNumber": 32767,
        "Protocol": "-1",
        "Egress": true,
        "RuleAction": "deny"
      },
      {
        "CidrBlock": "0.0.0.0/0",
        "RuleNumber": 32767,
        "Protocol": "-1",
        "Egress": false,
        "RuleAction": "deny"
      }
    ],
    "IsDefault": false
  }
}
```

- API-Einheiten finden Sie [CreateNetworkAcl](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Netzwerk ACL für das angegebene Objekt erstellt VPC.

```
New-EC2NetworkAcl -VpcId vpc-12345678
```

Ausgabe:

```
Associations : {}
Entries      : {Amazon.EC2.Model.NetworkAclEntry,
               Amazon.EC2.Model.NetworkAclEntry}
IsDefault   : False
NetworkAclId : acl-12345678
Tags        : {}
VpcId       : vpc-12345678
```

- API-Einheiten finden Sie unter [CreateNetworkAcl AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **CreateNetworkAclEntry** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CreateNetworkAclEntry`.

CLI

AWS CLI

Um einen ACL Netzwerkeintrag zu erstellen

In diesem Beispiel wird ein Eintrag für das angegebene Netzwerk erstellt ACL. Die Regel erlaubt eingehenden Datenverkehr von einer beliebigen IPv4 Adresse (0.0.0.0/0) an UDP Port 53 (DNS) in jedes zugehörige Subnetz. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 create-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-
number 100 --protocol udp --port-range From=53,To=53 --cidr-block 0.0.0.0/0 --
rule-action allow
```

In diesem Beispiel wird eine Regel für das angegebene Netzwerk erstelltACL, die eingehenden Datenverkehr von einer beliebigen IPv6 Adresse (:: /0) an Port 80 () zulässt. TCP HTTP

Befehl:

```
aws ec2 create-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-
number 120 --protocol tcp --port-range From=80,To=80 --ipv6-cidr-block ::/0 --
rule-action allow
```

- APIEinzelheiten finden Sie unter [CreateNetworkAclEntry AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Eintrag für das angegebene Netzwerk erstelltACL. Die Regel erlaubt eingehenden Verkehr von überall (0.0.0.0/0) auf UDP Port 53 (DNS) in jedes zugehörige Subnetz.

```
New-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber
100 -Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 0.0.0.0/0 -
RuleAction allow
```

- APIEinzelheiten finden Sie unter Cmdlet-Referenz. [CreateNetworkAclEntry](#) AWS Tools for PowerShell

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter. [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **CreateNetworkInterface** mit einem CLI

Die folgenden Codebeispiele zeigen die VerwendungCreateNetworkInterface.

CLI

AWS CLI

Beispiel 1: Um eine IPv4 Adresse für eine Netzwerkschnittstelle anzugeben

Im folgenden `create-network-interface` Beispiel wird eine Netzwerkschnittstelle für das angegebene Subnetz mit der angegebenen primären IPv4 Adresse erstellt.

```
aws ec2 create-network-interface \  
  --subnet-id subnet-00a24d0d67acf6333 \  
  --description "my network interface" \  
  --groups sg-09dfba7ed20cda78b \  
  --private-ip-address 10.0.8.17
```

Ausgabe:

```
{  
  "NetworkInterface": {  
    "AvailabilityZone": "us-west-2a",  
    "Description": "my network interface",  
    "Groups": [  
      {  
        "GroupName": "my-security-group",  
        "GroupId": "sg-09dfba7ed20cda78b"  
      }  
    ],  
    "InterfaceType": "interface",  
    "Ipv6Addresses": [],  
    "MacAddress": "06:6a:0f:9a:49:37",  
    "NetworkInterfaceId": "eni-0492b355f0cf3b3f8",  
    "OwnerId": "123456789012",  
    "PrivateDnsName": "ip-10-0-8-18.us-west-2.compute.internal",  
    "PrivateIpAddress": "10.0.8.17",  
    "PrivateIpAddresses": [  
      {  
        "Primary": true,  
        "PrivateDnsName": "ip-10-0-8-17.us-west-2.compute.internal",  
        "PrivateIpAddress": "10.0.8.17"  
      }  
    ],  
    "RequesterId": "AIDA4Z3Y7GSXTMEXAMPLE",  
    "RequesterManaged": false,  
  }  
}
```

```

    "SourceDestCheck": true,
    "Status": "pending",
    "SubnetId": "subnet-00a24d0d67acf6333",
    "TagSet": [],
    "VpcId": "vpc-02723a0feeeb9d57b"
  }
}

```

Beispiel 2: Um eine Netzwerkschnittstelle mit einer IPv4 Adresse und einer IPv6 Adresse zu erstellen

Das folgende `create-network-interface` Beispiel erstellt eine Netzwerkschnittstelle für das angegebene Subnetz mit einer IPv4 Adresse und einer IPv6 Adresse, die von Amazon EC2 ausgewählt wurden.

```

aws ec2 create-network-interface \
  --subnet-id subnet-00a24d0d67acf6333 \
  --description "my dual stack network interface" \
  --ipv6-address-count 1 \
  --groups sg-09dfba7ed20cda78b

```

Ausgabe:

```

{
  "NetworkInterface": {
    "AvailabilityZone": "us-west-2a",
    "Description": "my dual stack network interface",
    "Groups": [
      {
        "GroupName": "my-security-group",
        "GroupId": "sg-09dfba7ed20cda78b"
      }
    ],
    "InterfaceType": "interface",
    "Ipv6Addresses": [
      {
        "Ipv6Address": "2600:1f13:cfe:3650:a1dc:237c:393a:4ba7",
        "IsPrimaryIpv6": false
      }
    ],
    "MacAddress": "06:b8:68:d2:b2:2d",
    "NetworkInterfaceId": "eni-05da417453f9a84bf",
  }
}

```

```

    "OwnerId": "123456789012",
    "PrivateDnsName": "ip-10-0-8-18.us-west-2.compute.internal",
    "PrivateIpAddress": "10.0.8.18",
    "PrivateIpAddresses": [
      {
        "Primary": true,
        "PrivateDnsName": "ip-10-0-8-18.us-west-2.compute.internal",
        "PrivateIpAddress": "10.0.8.18"
      }
    ],
    "RequesterId": "AIDA4Z3Y7GSXTMEXAMPLE",
    "RequesterManaged": false,
    "SourceDestCheck": true,
    "Status": "pending",
    "SubnetId": "subnet-00a24d0d67acf6333",
    "TagSet": [],
    "VpcId": "vpc-02723a0feeeb9d57b",
    "Ipv6Address": "2600:1f13:cfe:3650:a1dc:237c:393a:4ba7"
  }
}

```

Beispiel 3: So erstellen Sie eine Netzwerkschnittstelle mit Konfigurationsoptionen für die Verbindungsverfolgung

Im folgenden `create-network-interface` Beispiel wird eine Netzwerkschnittstelle erstellt und die Timeouts für die Verbindungsverfolgung im Leerlauf konfiguriert.

```

aws ec2 create-network-interface \
  --subnet-id subnet-00a24d0d67acf6333 \
  --groups sg-02e57dbcfe0331c1b \
  --connection-tracking-specification TcpEstablishedTimeout=86400,UdpTimeout=60

```

Ausgabe:

```

{
  "NetworkInterface": {
    "AvailabilityZone": "us-west-2a",
    "ConnectionTrackingConfiguration": {
      "TcpEstablishedTimeout": 86400,
      "UdpTimeout": 60
    },
    "Description": ""
  }
}

```

```

    "Groups": [
      {
        "GroupName": "my-security-group",
        "GroupId": "sg-02e57dbcfe0331c1b"
      }
    ],
    "InterfaceType": "interface",
    "Ipv6Addresses": [],
    "MacAddress": "06:4c:53:de:6d:91",
    "NetworkInterfaceId": "eni-0c133586e08903d0b",
    "OwnerId": "123456789012",
    "PrivateDnsName": "ip-10-0-8-94.us-west-2.compute.internal",
    "PrivateIpAddress": "10.0.8.94",
    "PrivateIpAddresses": [
      {
        "Primary": true,
        "PrivateDnsName": "ip-10-0-8-94.us-west-2.compute.internal",
        "PrivateIpAddress": "10.0.8.94"
      }
    ],
    "RequesterId": "AIDA4Z3Y7GSXTMEXAMPLE",
    "RequesterManaged": false,
    "SourceDestCheck": true,
    "Status": "pending",
    "SubnetId": "subnet-00a24d0d67acf6333",
    "TagSet": [],
    "VpcId": "vpc-02723a0feeeb9d57b"
  }
}

```

Beispiel 4: So erstellen Sie einen Elastic Fabric-Adapter

Das folgende `create-network-interface` Beispiel erstellt eine EFA.

```

aws ec2 create-network-interface \
  --interface-type efa \
  --subnet-id subnet-00a24d0d67acf6333 \
  --description "my efa" \
  --groups sg-02e57dbcfe0331c1b

```

Ausgabe:

```
{
```

```

"NetworkInterface": {
  "AvailabilityZone": "us-west-2a",
  "Description": "my efa",
  "Groups": [
    {
      "GroupName": "my-efa-sg",
      "GroupId": "sg-02e57dbcfe0331c1b"
    }
  ],
  "InterfaceType": "efa",
  "Ipv6Addresses": [],
  "MacAddress": "06:d7:a4:f7:4d:57",
  "NetworkInterfaceId": "eni-034acc2885e862b65",
  "OwnerId": "123456789012",
  "PrivateDnsName": "ip-10-0-8-180.us-west-2.compute.internal",
  "PrivateIpAddress": "10.0.8.180",
  "PrivateIpAddresses": [
    {
      "Primary": true,
      "PrivateDnsName": "ip-10-0-8-180.us-west-2.compute.internal",
      "PrivateIpAddress": "10.0.8.180"
    }
  ],
  "RequesterId": "AIDA4Z3Y7GSXTMEXAMPLE",
  "RequesterManaged": false,
  "SourceDestCheck": true,
  "Status": "pending",
  "SubnetId": "subnet-00a24d0d67acf6333",
  "TagSet": [],
  "VpcId": "vpc-02723a0feeeb9d57b"
}
}

```

Weitere Informationen finden Sie unter [Elastic Network Interfaces](#) im EC2Amazon-Benutzerhandbuch.

- API-Einheiten finden Sie [CreateNetworkInterface](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle erstellt.


```
New-EC2NetworkInterface -SubnetId subnet-1a2b3c4d -Description "my network interface" -Group sg-12345678 -PrivateIpAddress 10.0.0.17
```

Ausgabe:

```
Association      :
Attachment      :
AvailabilityZone : us-west-2c
Description     : my network interface
Groups         : {my-security-group}
MacAddress      : 0a:72:bc:1a:cd:7f
NetworkInterfaceId : eni-12345678
OwnerId        : 123456789012
PrivateDnsName  : ip-10-0-0-17.us-west-2.compute.internal
PrivateIpAddress : 10.0.0.17
PrivateIpAddresses : {}
RequesterId    :
RequesterManaged : False
SourceDestCheck : True
Status         : pending
SubnetId       : subnet-1a2b3c4d
TagSet         : {}
VpcId         : vpc-12345678
```

- API-Einheiten finden Sie unter [CreateNetworkInterface AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **CreatePlacementGroup** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CreatePlacementGroup`.

CLI

AWS CLI

Um eine Platzierungsgruppe zu erstellen

Dieser Beispielbefehl erstellt eine Platzierungsgruppe mit dem angegebenen Namen.

Befehl:

```
aws ec2 create-placement-group --group-name my-cluster --strategy cluster
```

Um eine Platzierungsgruppe für Partitionen zu erstellen

Dieser Beispielbefehl erstellt eine Partitionsplatzierungsgruppe HDFS-Group-A mit dem Namen fünf Partitionen.

Befehl:

```
aws ec2 create-placement-group --group-name HDFS-Group-A --strategy partition --partition-count 5
```

- API-Einheiten finden Sie [CreatePlacementGroup](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Platzierungsgruppe mit dem angegebenen Namen erstellt.

```
New-EC2PlacementGroup -GroupName my-placement-group -Strategy cluster
```

- API-Einheiten finden Sie unter [CreatePlacementGroup AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **CreateRoute** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CreateRoute`.

CLI

AWS CLI

Um eine Route zu erstellen

In diesem Beispiel wird eine Route für die angegebene Routentabelle erstellt. Die Route entspricht dem gesamten IPv4 Verkehr ($0.0.0.0/0$) und leitet ihn an das angegebene Internet-Gateway weiter. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 create-route --route-table-id rtb-22574640 --destination-cidr-block 0.0.0.0/0 --gateway-id igw-c0a643a9
```

Dieser Beispielbefehl erstellt eine Route in der Routentabelle `rtb-g8ff4ea2`. Die Route entspricht dem Verkehr für den IPv4 CIDR Block `10.0.0.0/16` und leitet ihn an die Peering-Verbindung `pcx-111aaa22` weiter. VPC Diese Route ermöglicht die Weiterleitung des Datenverkehrs an den Peer in der Peering-Verbindung. VPC Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 create-route --route-table-id rtb-g8ff4ea2 --destination-cidr-block 10.0.0.0/16 --vpc-peering-connection-id pcx-1a2b3c4d
```

In diesem Beispiel wird in der angegebenen Routentabelle eine Route erstellt, die dem gesamten IPv6 Datenverkehr ($:::/0$) entspricht, und ihn an das angegebene Internet-Gateway weiterleitet, das nur für ausgehenden Datenverkehr bestimmt ist.

Befehl:

```
aws ec2 create-route --route-table-id rtb-dce620b8 --destination-ipv6-cidr-block :::/0 --egress-only-internet-gateway-id eigw-01eadbd45ecd7943f
```

- API-Einheiten finden Sie unter Befehlsreferenz [CreateRoute](#).AWS CLI

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Route für die angegebene Routentabelle erstellt. Die Route entspricht dem gesamten Datenverkehr und sendet ihn an das angegebene Internet-Gateway.

```
New-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0 -  
GatewayId igw-1a2b3c4d
```

Ausgabe:

```
True
```

- API-Einheiten finden Sie unter [CreateRoute AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie es **CreateRouteTable** mit einem AWS SDK oder CLI.

Die folgenden Codebeispiele zeigen, wie man es benutzt `CreateRouteTable`.

CLI

AWS CLI

So erstellen Sie eine Routing-Tabelle.

In diesem Beispiel wird eine Routentabelle für das angegebene Objekt erstellt `VPC`.

Befehl:

```
aws ec2 create-route-table --vpc-id vpc-a01106c2
```

Ausgabe:

```
{  
  "RouteTable": {  
    "Associations": [],  
    "RouteTableId": "rtb-22574640",  
    "VpcId": "vpc-a01106c2",  
    "PropagatingVgws": [],  
    "Tags": [],  
    "Routes": [  
      {  
        "GatewayId": "local",
```

```

        "DestinationCidrBlock": "10.0.0.0/16",
        "State": "active"
      }
    ]
  }
}

```

- API-Einheiten finden Sie [CreateRouteTable](#) unter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Routentabelle für das angegebene Objekt erstellt VPC.

```
New-EC2RouteTable -VpcId vpc-12345678
```

Ausgabe:

```

Associations      : {}
PropagatingVgws  : {}
Routes           : {}
RouteTableId     : rtb-1a2b3c4d
Tags             : {}
VpcId            : vpc-12345678

```

- API-Einheiten finden Sie unter [CreateRouteTable AWS Tools for PowerShell Cmdlet](#)-Referenz.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
require 'aws-sdk-ec2'
```

```
# Prerequisites:
#
# - A VPC in Amazon VPC.
# - A subnet in that VPC.
# - A gateway attached to that subnet.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the route table.
# @param subnet_id [String] The ID of the subnet for the route table.
# @param gateway_id [String] The ID of the gateway for the route.
# @param destination_cidr_block [String] The destination CIDR block
#   for the route.
# @param tag_key [String] The key portion of the tag for the route table.
# @param tag_value [String] The value portion of the tag for the route table.
# @return [Boolean] true if the route table was created and associated;
#   otherwise, false.
# @example
#   exit 1 unless route_table_created_and_associated?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-0b6f769731EXAMPLE',
#     'subnet-03d9303b57EXAMPLE',
#     'igw-06ca90c011EXAMPLE',
#     '0.0.0.0/0',
#     'my-key',
#     'my-value'
#   )
def route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  route_table = ec2_resource.create_route_table(vpc_id: vpc_id)
  puts "Created route table with ID '#{route_table.id}'."
  route_table.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
end
```

```

    }
  ]
)
puts 'Added tags to route table.'
route_table.create_route(
  destination_cidr_block: destination_cidr_block,
  gateway_id: gateway_id
)
puts 'Created route with destination CIDR block ' \
    "'#{destination_cidr_block}' and associated with gateway " \
    "with ID '#{gateway_id}'."
route_table.associate_with_subnet(subnet_id: subnet_id)
puts "Associated route table with subnet with ID '#{subnet_id}'."
true
rescue StandardError => e
  puts "Error creating or associating route table: #{e.message}"
  puts 'If the route table was created but not associated, you should ' \
    'clean up by deleting the route table.'
  false
end

# Example usage:
def run_me
  vpc_id = ''
  subnet_id = ''
  gateway_id = ''
  destination_cidr_block = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage: ruby ec2-ruby-example-create-route-table.rb ' \
        'VPC_ID SUBNET_ID GATEWAY_ID DESTINATION_CIDR_BLOCK ' \
        'TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-route-table.rb ' \
        'vpc-0b6f769731EXAMPLE subnet-03d9303b57EXAMPLE igw-06ca90c011EXAMPLE ' \
        "'0.0.0.0/0' my-key my-value us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = 'vpc-0b6f769731EXAMPLE'
    subnet_id = 'subnet-03d9303b57EXAMPLE'

```

```
gateway_id = 'igw-06ca90c011EXAMPLE'
destination_cidr_block = '0.0.0.0/0'
tag_key = 'my-key'
tag_value = 'my-value'
# Replace us-west-2 with the AWS Region you're using for Amazon EC2.
region = 'us-west-2'
# Otherwise, use the values as specified at the command prompt.
else
  vpc_id = ARGV[0]
  subnet_id = ARGV[1]
  gateway_id = ARGV[2]
  destination_cidr_block = ARGV[3]
  tag_key = ARGV[4]
  tag_value = ARGV[5]
  region = ARGV[6]
end

ec2_resource = Aws::EC2::Resource.new(region: region)

if route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  puts 'Route table created and associated.'
else
  puts 'Route table not created or not associated.'
end
end

run_me if $PROGRAM_NAME == __FILE__
```

- API-Einheiten finden Sie [CreateRouteTable](#) unter AWS SDK for Ruby API-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie es **CreateSecurityGroup** mit einem AWS SDK oder CLI


Die folgenden Codebeispiele zeigen, wie man es benutzt `CreateSecurityGroup`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)

.NET

AWS SDK for .NET

 Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Create an Amazon EC2 security group with a specified name and
description.
/// </summary>
/// <param name="groupName">The name for the new security group.</param>
/// <param name="groupDescription">A description of the new security group.</
param>
/// <returns>The group Id of the new security group.</returns>
public async Task<string> CreateSecurityGroup(string groupName, string
groupDescription)
{
    try
    {
        var response = await _amazonEC2.CreateSecurityGroupAsync(
            new CreateSecurityGroupRequest(groupName, groupDescription));

        // Wait until the security group exists.
        int retries = 5;
```


```
        while (retries-- > 0)
        {
            var groups = await DescribeSecurityGroups(response.GroupId);
            if (groups.Any())
            {
                return response.GroupId;
            }

            Thread.Sleep(5000);
            retries--;
        }
        _logger.LogError($"Unable to find newly created group {groupName}.");
        throw new DoesNotExistException("security group not found");
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "ResourceAlreadyExists")
        {
            _logger.LogError(
                $"A security group with the name {groupName} already exists.
{ec2Exception.Message}");
        }
        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(
            $"An error occurred while creating the security group.:
{ex.Message}");
        throw;
    }
}
```

- API-Einheiten finden Sie [CreateSecurityGroup](#) unter AWS SDK for .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

 Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#####
# function ec2_create_security_group
#
# This function creates an Amazon Elastic Compute Cloud (Amazon EC2) security
# group.
#
# Parameters:
#     -n security_group_name - The name of the security group.
#     -d security_group_description - The description of the security group.
#
# Returns:
#     The ID of the created security group, or an error message if the
#     operation fails.
# And:
#     0 - If successful.
#     1 - If it fails.
#
#####
function ec2_create_security_group() {
    local security_group_name security_group_description response

    # Function to display usage information
    function usage() {
        echo "function ec2_create_security_group"
        echo "Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group."
        echo "  -n security_group_name - The name of the security group."
        echo "  -d security_group_description - The description of the security
group."
        echo ""
    }

    # Parse the command-line arguments
```

```
while getopts "n:d:h" option; do
  case "${option}" in
    n) security_group_name="${OPTARG}" ;;
    d) security_group_description="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

# Validate the input parameters
if [[ -z "$security_group_name" ]]; then
  errecho "ERROR: You must provide a security group name with the -n
parameter."
  return 1
fi

if [[ -z "$security_group_description" ]]; then
  errecho "ERROR: You must provide a security group description with the -d
parameter."
  return 1
fi

# Create the security group
response=$(aws ec2 create-security-group \
  --group-name "$security_group_name" \
  --description "$security_group_description" \
  --query "GroupId" \
  --output text) || {
  aws_cli_error_log ${?}
  errecho "ERROR: AWS reports create-security-group operation failed."
  errecho "$response"
  return 1
}

echo "$response"
return 0
```

```
}

```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then
        errecho " The service returned an error."
    elif [ "$err_code" == 255 ]; then
        errecho " 255 is a catch-all error."
    }
}

```

```

fi

return 0
}

```

- API-Einheiten finden Sie [CreateSecurityGroup](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

//! Create a security group.
/!*
 \param groupName: A security group name.
 \param description: A description.
 \param vpcID: A virtual private cloud (VPC) ID.
 \param[out] groupIDResult: A string to receive the group ID.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::EC2::createSecurityGroup(const Aws::String &groupName,
                                     const Aws::String &description,
                                     const Aws::String &vpcID,
                                     Aws::String &groupIDResult,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::CreateSecurityGroupRequest request;

    request.SetGroupName(groupName);
    request.SetDescription(description);
    request.SetVpcId(vpcID);

    const Aws::EC2::Model::CreateSecurityGroupOutcome outcome =

```

```
        ec2Client.CreateSecurityGroup(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to create security group:" <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    std::cout << "Successfully created security group named " << groupName <<
        std::endl;

    groupIDResult = outcome.GetResult().GetGroupId();

    return true;
}
```

- API-Einheiten finden Sie [CreateSecurityGroup](#) unter AWS SDK for C++ API-Referenz.

CLI

AWS CLI

Um eine Sicherheitsgruppe für EC2 -Classic zu erstellen

In diesem Beispiel wird eine Sicherheitsgruppe mit dem Namen MySecurityGroup erstellt.

Befehl:

```
aws ec2 create-security-group --group-name MySecurityGroup --description "My security group"
```

Ausgabe:

```
{
  "GroupId": "sg-903004f8"
}
```

Um eine Sicherheitsgruppe für EC2 - zu erstellen VPC

In diesem Beispiel wird eine Sicherheitsgruppe erstellt, die MySecurityGroup nach der angegebenen benannt ist VPC.

Befehl:

```
aws ec2 create-security-group --group-name MySecurityGroup --description "My security group" --vpc-id vpc-1a2b3c4d
```

Ausgabe:

```
{
  "GroupId": "sg-903004f8"
}
```

Weitere Informationen finden Sie unter Verwenden von Sicherheitsgruppen im Benutzerhandbuch für die AWS -Befehlszeilenschnittstelle.

- API-Einheiten finden Sie [CreateSecurityGroup](#) unter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Creates a new security group asynchronously with the specified group name,
 * description, and VPC ID. It also
 *   * authorizes inbound traffic on ports 80 and 22 from the specified IP
 * address.
 *
 * @param groupName the name of the security group to create
 * @param groupDesc the description of the security group
 * @param vpcId the ID of the VPC in which to create the security
 * group
 * @param myIpAddress the IP address from which to allow inbound traffic
 * (e.g., "192.168.1.1/0" to allow traffic from
```



```

    *          any IP address in the 192.168.1.0/24 subnet)
    * @return a CompletableFuture that, when completed, returns the ID of the
    * created security group
    * @throws RuntimeException if there was a failure creating the security
    * group or authorizing the inbound traffic
    */
    public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

        return getAsyncClient().createSecurityGroup(createRequest)
            .thenCompose(createResponse -> {
                String groupId = createResponse.groupId();
                IpRange ipRange = IpRange.builder()
                    .cidrIp(myIpAddress + "/32")
                    .build();

                IpPermission ipPerm = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(80)
                    .fromPort(80)
                    .ipRanges(ipRange)
                    .build();

                IpPermission ipPerm2 = IpPermission.builder()
                    .ipProtocol("tcp")
                    .toPort(22)
                    .fromPort(22)
                    .ipRanges(ipRange)
                    .build();

                AuthorizeSecurityGroupIngressRequest authRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                    .groupName(groupName)
                    .ipPermissions(ipPerm, ipPerm2)
                    .build();

                return
getAsyncClient().authorizeSecurityGroupIngress(authRequest)

```

```

        .thenApply(authResponse -> groupId);
    })
    .whenComplete((result, exception) -> {
        if (exception != null) {
            if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                throw (Ec2Exception) exception.getCause();
            } else {
                throw new RuntimeException("Failed to create security
group: " + exception.getMessage(), exception);
            }
        }
    });
}

```

- API-Einheiten finden Sie [CreateSecurityGroup](#) unter AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import { CreateSecurityGroupCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Creates a security group.
 * @param {{ groupName: string, description: string }} options
 */
export const main = async ({ groupName, description }) => {
    const client = new EC2Client({});
    const command = new CreateSecurityGroupCommand({
        // Up to 255 characters in length. Cannot start with sg-.
        GroupName: groupName,
        // Up to 255 characters in length.
        Description: description,
    });
};

```

```
try {
    const { GroupId } = await client.send(command);
    console.log(GroupId);
} catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
        console.warn(`${caught.message}.`);
    } else {
        throw caught;
    }
}
};
```

- API-Einheiten finden Sie [CreateSecurityGroup](#) unter AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
suspend fun createEC2SecurityGroup(
    groupNameVal: String?,
    groupDescVal: String?,
    vpcIdVal: String?,
): String? {
    val request =
        CreateSecurityGroupRequest {
            groupName = groupNameVal
            description = groupDescVal
            vpcId = vpcIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val resp = ec2.createSecurityGroup(request)
    }
}
```

```
    val ipRange =
        IpRange {
            cidrIp = "0.0.0.0/0"
        }

    val ipPerm =
        IpPermission {
            ipProtocol = "tcp"
            toPort = 80
            fromPort = 80
            ipRanges = listOf(ipRange)
        }

    val ipPerm2 =
        IpPermission {
            ipProtocol = "tcp"
            toPort = 22
            fromPort = 22
            ipRanges = listOf(ipRange)
        }

    val authRequest =
        AuthorizeSecurityGroupIngressRequest {
            groupName = groupNameVal
            ipPermissions = listOf(ipPerm, ipPerm2)
        }
    ec2.authorizeSecurityGroupIngress(authRequest)
    println("Successfully added ingress policy to Security Group
    $groupNameVal")
    return resp.groupId
}
}
```

- API-Einheiten finden Sie [CreateSecurityGroup](#) in der AWS SDK-API-Kotlin-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Sicherheitsgruppe für die angegebene Gruppe erstelltVPC.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group" -VpcId vpc-12345678
```

Ausgabe:

```
sg-12345678
```

Beispiel 2: In diesem Beispiel wird eine Sicherheitsgruppe für EC2 -Classic erstellt.

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security group"
```

Ausgabe:

```
sg-45678901
```

- API-Einheiten finden Sie unter [CreateSecurityGroup AWS Tools for PowerShell Cmdlet-Referenz](#).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] = None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional security group ID.
        """
```

```
        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
        access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
high-level identifier
        that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def create(self, group_name: str, group_description: str) -> str:
        """
        Creates a security group in the default virtual private cloud (VPC) of
the current account.

        :param group_name: The name of the security group to create.
        :param group_description: The description of the security group to
create.
        :return: The ID of the newly created security group.
        :raise Handles AWS SDK service-level ClientError, with special handling
for ResourceAlreadyExists
        """
        try:
            response = self.ec2_client.create_security_group(
                GroupName=group_name, Description=group_description
            )
            self.security_group = response["GroupId"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceAlreadyExists":
                logger.error(
                    f"Security group '{group_name}' already exists. Please choose
a different name."
```

```

        )
        raise
    else:
        return self.security_group

```

- API-Einheiten finden Sie unter [CreateSecurityGroup](#) Python (Boto3) API -Referenz. AWS SDK

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

# This code example does the following:
# 1. Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
# 2. Adds inbound rules to the security group.
# 3. Displays information about available security groups.
# 4. Deletes the security group.

require 'aws-sdk-ec2'

# Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Prerequisites:
#
# - A VPC in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param group_name [String] A name for the security group.
# @param description [String] A description for the security group.
# @param vpc_id [String] The ID of the VPC for the security group.
# @return [String] The ID of security group that was created.

```

```
# @example
#   puts create_security_group(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-security-group',
#     'This is my security group.',
#     'vpc-6713dfEX'
#   )
def create_security_group(ec2_client, group_name, description, vpc_id)
  security_group = ec2_client.create_security_group(
    group_name: group_name,
    description: description,
    vpc_id: vpc_id
  )
  puts "Created security group '#{group_name}' with ID " \
    "'#{security_group.group_id}' in VPC with ID '#{vpc_id}'."
  security_group.group_id
rescue StandardError => e
  puts "Error creating security group: #{e.message}"
  'Error'
end

# Adds an inbound rule to an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
#
# Prerequisites:
#
# - The security group.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param security_group_id [String] The ID of the security group.
# @param ip_protocol [String] The network protocol for the inbound rule.
# @param from_port [String] The originating port for the inbound rule.
# @param to_port [String] The destination port for the inbound rule.
# @param cidr_ip_range [String] The CIDR IP range for the inbound rule.
# @return
# @example
#   exit 1 unless security_group_ingress_authorized?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'sg-030a858e078f1b9EX',
#     'tcp',
#     '80',
#     '80',
#     '0.0.0.0/0'
#   )
```



```

def security_group_ingress_authorized?(
  ec2_client, security_group_id, ip_protocol, from_port, to_port, cidr_ip_range
)
  ec2_client.authorize_security_group_ingress(
    group_id: security_group_id,
    ip_permissions: [
      {
        ip_protocol: ip_protocol,
        from_port: from_port,
        to_port: to_port,
        ip_ranges: [
          {
            cidr_ip: cidr_ip_range
          }
        ]
      }
    ]
  )
  puts "Added inbound rule to security group '#{security_group_id}' for protocol
" \
  "'#{ip_protocol}' from port '#{from_port}' to port '#{to_port}' " \
  "with CIDR IP range '#{cidr_ip_range}'."
  true
rescue StandardError => e
  puts "Error adding inbound rule to security group: #{e.message}"
  false
end

# Refactored method to simplify complexity for describing security group
permissions
def format_port_information(perm)
  from_port_str = perm.from_port == '-1' || perm.from_port == -1 ? 'All' :
perm.from_port.to_s
  to_port_str = perm.to_port == '-1' || perm.to_port == -1 ? 'All' :
perm.to_port.to_s
  { from_port: from_port_str, to_port: to_port_str }
end

# Displays information about a security group's IP permissions set in
# Amazon Elastic Compute Cloud (Amazon EC2).
def describe_security_group_permissions(perm)
  ports = format_port_information(perm)

  print " Protocol: #{perm.ip_protocol == '-1' ? 'All' : perm.ip_protocol}"

```

```
print ", From: #{ports[:from_port]}, To: #{ports[:to_port]}"

print ", CIDR IPv6: #{perm.ipv_6_ranges[0].cidr_ipv_6}" if perm.key?
(:ipv_6_ranges) && perm.ipv_6_ranges.count.positive?

print ", CIDR IPv4: #{perm.ip_ranges[0].cidr_ip}" if perm.key?(:ip_ranges) &&
perm.ip_ranges.count.positive?
print "\n"
end

# Displays information about available security groups in
# Amazon Elastic Compute Cloud (Amazon EC2).
def describe_security_groups(ec2_client)
  response = ec2_client.describe_security_groups

  if response.security_groups.count.positive?
    response.security_groups.each do |sg|
      display_group_details(sg)
    end
  else
    puts 'No security groups found.'
  end
rescue StandardError => e
  puts "Error getting information about security groups: #{e.message}"
end

# Helper method to display the details of security groups
def display_group_details(sg)
  puts '-' * (sg.group_name.length + 13)
  puts "Name:      #{sg.group_name}"
  puts "Description: #{sg.description}"
  puts "Group ID:   #{sg.group_id}"
  puts "Owner ID:   #{sg.owner_id}"
  puts "VPC ID:     #{sg.vpc_id}"

  display_group_tags(sg.tags) if sg.tags.count.positive?
  display_group_permissions(sg)
end

def display_group_tags(tags)
  puts 'Tags:'
  tags.each do |tag|
    puts "  Key: #{tag.key}, Value: #{tag.value}"
  end
end
```

```
end

def display_group_permissions(sg)
  if sg.ip_permissions.count.positive?
    puts 'Inbound rules:'
    sg.ip_permissions.each do |p|
      describe_security_group_permissions(p)
    end
  end
end

return if sg.ip_permissions_egress.empty?

puts 'Outbound rules:'
sg.ip_permissions_egress.each do |p|
  describe_security_group_permissions(p)
end
end

# Deletes an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
def security_group_deleted?(ec2_client, security_group_id)
  ec2_client.delete_security_group(group_id: security_group_id)
  puts "Deleted security group '#{security_group_id}'."
  true
rescue StandardError => e
  puts "Error deleting security group: #{e.message}"
  false
end

# Example usage with refactored run_me to reduce complexity
def run_me
  group_name, description, vpc_id, ip_protocol_http, from_port_http,
  to_port_http, \
  cidr_ip_range_http, ip_protocol_ssh, from_port_ssh, to_port_ssh, \
  cidr_ip_range_ssh, region = process_arguments
  ec2_client = Aws::EC2::Client.new(region: region)

  security_group_id = attempt_create_security_group(ec2_client, group_name,
description, vpc_id)
  security_group_exists = security_group_id != 'Error'

  if security_group_exists
    add_inbound_rules(ec2_client, security_group_id, ip_protocol_http,
from_port_http, to_port_http, cidr_ip_range_http)
```

```
    add_inbound_rules(ec2_client, security_group_id, ip_protocol_ssh,
from_port_ssh, to_port_ssh, cidr_ip_range_ssh)
  end

  describe_security_groups(ec2_client)
  attempt_delete_security_group(ec2_client, security_group_id) if
security_group_exists
end

def process_arguments
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    display_help
    exit 1
  elsif ARGV.count.zero?
    default_values
  else
    ARGV
  end
end

def attempt_create_security_group(ec2_client, group_name, description, vpc_id)
  puts 'Attempting to create security group...'
  security_group_id = create_security_group(ec2_client, group_name, description,
vpc_id)
  puts 'Could not create security group. Skipping this step.' if
security_group_id == 'Error'
  security_group_id
end

def add_inbound_rules(ec2_client, security_group_id, ip_protocol, from_port,
to_port, cidr_ip_range)
  puts 'Attempting to add inbound rules to security group...'
  return if security_group_ingress_authorized?(ec2_client, security_group_id,
ip_protocol, from_port, to_port,
                                         cidr_ip_range)

  puts 'Could not add inbound rule to security group. Skipping this step.'
end

def attempt_delete_security_group(ec2_client, security_group_id)
  puts "\nAttempting to delete security group..."
  return if security_group_deleted?(ec2_client, security_group_id)

  puts 'Could not delete security group. You must delete it yourself.'
```

```

end

def display_help
  puts 'Usage:  ruby ec2-ruby-example-security-group.rb ' \
        'GROUP_NAME DESCRIPTION VPC_ID IP_PROTOCOL_1 FROM_PORT_1 TO_PORT_1 ' \
        'CIDR_IP_RANGE_1 IP_PROTOCOL_2 FROM_PORT_2 TO_PORT_2 ' \
        'CIDR_IP_RANGE_2 REGION'
  puts 'Example: ruby ec2-ruby-example-security-group.rb ' \
        "my-security-group 'This is my security group.' vpc-6713dfEX " \
        "tcp 80 80 '0.0.0.0/0' tcp 22 22 '0.0.0.0/0' us-west-2"
end

def default_values
  [
    'my-security-group', 'This is my security group.', 'vpc-6713dfEX', 'tcp',
    '80', '80',
    '0.0.0.0/0', 'tcp', '22', '22', '0.0.0.0/0', 'us-west-2'
  ]
end

run_me if $PROGRAM_NAME == __FILE__

```

- API-Einheiten finden Sie [CreateSecurityGroup](#) unter AWS SDK for Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

pub async fn create_security_group(
    &self,
    name: &str,
    description: &str,
) -> Result<SecurityGroup, EC2Error> {
    tracing::info!("Creating security group {name}");
    let create_output = self

```

```

        .client
        .create_security_group()
        .group_name(name)
        .description(description)
        .send()
        .await
        .map_err(EC2Error::from)?;

    let group_id = create_output
        .group_id
        .ok_or_else(|| EC2Error::new("Missing security group id after
creation"))?;

    let group = self
        .describe_security_group(&group_id)
        .await?
        .ok_or_else(|| {
            EC2Error::new(format!("Could not find security group with id
{group_id}"))
        })?;

    tracing::info!("Created security group {name} as {group_id}");

    Ok(group)
}

```

- API-Einheiten finden Sie [CreateSecurityGroup](#) in der AWS SDK-API-Rust-Referenz.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

TRY.

```

    oo_result = lo_ec2->createsecuritygroup(
        " oo_result is
returned for testing purposes. "

```

```

        iv_description = 'Security group example'
        iv_groupname = iv_security_group_name
        iv_vpcid = iv_vpc_id
    ).
    MESSAGE 'Security group created.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- API-Einheiten finden Sie [CreateSecurityGroup](#) unter AWS SDKSAPABAPAPI als Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **CreateSnapshot** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CreateSnapshot`.

CLI

AWS CLI

Um einen Snapshot zu erstellen

Dieser Beispielbefehl erstellt einen Snapshot des Volumes mit der Volume-ID `vol-1234567890abcdef0` und einer kurzen Beschreibung zur Identifizierung des Snapshots.

Befehl:

```
aws ec2 create-snapshot --volume-id vol-1234567890abcdef0 --description "This is my root volume snapshot"
```

Ausgabe:

```
{
  "Description": "This is my root volume snapshot",
  "Tags": [],

```

```
"Encrypted": false,
"VolumeId": "vol-1234567890abcdef0",
"State": "pending",
"VolumeSize": 8,
"StartTime": "2018-02-28T21:06:01.000Z",
"Progress": "",
"OwnerId": "012345678910",
"SnapshotId": "snap-066877671789bd71b"
}
```

Um einen Snapshot mit Tags zu erstellen

Dieser Beispielbefehl erstellt einen Snapshot und wendet zwei Tags an: purpose=prod und costcenter=123.

Befehl:

```
aws ec2 create-snapshot --volume-id vol-1234567890abcdef0
--description 'Prod backup' --tag-specifications
'ResourceType=snapshot, Tags=[{Key=purpose, Value=prod},
{Key=costcenter, Value=123}]'
```

Ausgabe:

```
{
  "Description": "Prod backup",
  "Tags": [
    {
      "Value": "prod",
      "Key": "purpose"
    },
    {
      "Value": "123",
      "Key": "costcenter"
    }
  ],
  "Encrypted": false,
  "VolumeId": "vol-1234567890abcdef0",
  "State": "pending",
  "VolumeSize": 8,
  "StartTime": "2018-02-28T21:06:06.000Z",
  "Progress": "",
  "OwnerId": "012345678910",
```



```
"SnapshotId": "snap-09ed24a70bc19bbe4"  
}
```

- Einzelheiten API finden Sie in der Befehlsreferenz. [CreateSnapshot](#) AWS CLI

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Snapshot des angegebenen Volumes erstellt.

```
New-EC2Snapshot -VolumeId vol-12345678 -Description "This is a test"
```

Ausgabe:

```
DataEncryptionKeyId :  
Description          : This is a test  
Encrypted            : False  
KmsKeyId              :  
OwnerAlias           :  
OwnerId              : 123456789012  
Progress             :  
SnapshotId          : snap-12345678  
StartTime            : 12/22/2015 1:28:42 AM  
State                : pending  
StateMessage         :  
Tags                 : {}  
VolumeId             : vol-12345678  
VolumeSize           : 20
```

- API Einzelheiten finden Sie unter [CreateSnapshot AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter. [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **CreateSpotDatafeedSubscription** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CreateSpotDatafeedSubscription`.

CLI

AWS CLI

Um einen Spot-Instance-Datenfeed zu erstellen

Im folgenden `create-spot-datafeed-subscription` Beispiel wird ein Spot-Instance-Datenfeed erstellt.

```
aws ec2 create-spot-datafeed-subscription \  
  --bucket my-bucket \  
  --prefix spot-data-feed
```

Ausgabe:

```
{  
  "SpotDatafeedSubscription": {  
    "Bucket": "my-bucket",  
    "OwnerId": "123456789012",  
    "Prefix": "spot-data-feed",  
    "State": "Active"  
  }  
}
```

Der Datenfeed wird in dem von Ihnen angegebenen Amazon S3 S3-Bucket gespeichert. Die Dateinamen für diesen Datenfeed haben das folgende Format.

```
my-bucket.s3.amazonaws.com/spot-data-feed/123456789012.YYYY-MM-DD-  
HH.n.abcd1234.gz
```

Weitere Informationen finden Sie unter [Spot-Instance-Datenfeed](#) im Amazon Elastic Compute Cloud-Benutzerhandbuch für Linux-Instances.

- API-Einheiten finden Sie [CreateSpotDatafeedSubscription](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Spot-Instance-Datenfeed erstellt.

```
New-EC2SpotDatafeedSubscription -Bucket amzn-s3-demo-bucket -Prefix spotdata
```

Ausgabe:

```
Bucket   : my-s3-bucket
Fault    :
OwnerId  : 123456789012
Prefix   : spotdata
State    : Active
```

- API-Einheiten finden Sie unter [CreateSpotDatafeedSubscription AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie es **CreateSubnet** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `CreateSubnet`.

CLI

AWS CLI

Beispiel 1: Um ein Subnetz nur mit einem IPv4 CIDR-Block zu erstellen

Das folgende `create-subnet`-Beispiel erstellt ein Subnetz im angegebenen Bereich VPC mit dem angegebenen IPv4 CIDR-Block.

```
aws ec2 create-subnet \
  --vpc-id vpc-081ec835f3EXAMPLE \
  --cidr-block 10.0.0.0/24 \
  --tag-specifications ResourceType=subnet,Tags=[{Key=Name,Value=my-ipv4-only-subnet}]
```

Ausgabe:

```
{
  "Subnet": {
    "AvailabilityZone": "us-west-2a",
```

```

    "AvailabilityZoneId": "usw2-az2",
    "AvailableIpAddressCount": 251,
    "CidrBlock": "10.0.0.0/24",
    "DefaultForAz": false,
    "MapPublicIpOnLaunch": false,
    "State": "available",
    "SubnetId": "subnet-0e99b93155EXAMPLE",
    "VpcId": "vpc-081ec835f3EXAMPLE",
    "OwnerId": "123456789012",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [],
    "Tags": [
      {
        "Key": "Name",
        "Value": "my-ipv4-only-subnet"
      }
    ],
    "SubnetArn": "arn:aws:ec2:us-west-2:123456789012:subnet/
subnet-0e99b93155EXAMPLE"
  }
}

```

Beispiel 2: Um ein Subnetz mit Blöcken sowohl als auch IPv4 zu erstellen IPv6 CIDR

Im folgenden `create-subnet` Beispiel wird ein Subnetz im angegebenen Bereich VPC mit den angegebenen Blöcken IPv4 und IPv6 CIDR erstellt.

```

aws ec2 create-subnet \
  --vpc-id vpc-081ec835f3EXAMPLE \
  --cidr-block 10.0.0.0/24 \
  --ipv6-cidr-block 2600:1f16:cfe:3660::/64 \
  --tag-specifications ResourceType=subnet,Tags=[{Key=Name,Value=my-ipv4-ipv6-
subnet}]

```

Ausgabe:

```

{
  "Subnet": {
    "AvailabilityZone": "us-west-2a",
    "AvailabilityZoneId": "usw2-az2",
    "AvailableIpAddressCount": 251,
    "CidrBlock": "10.0.0.0/24",
    "DefaultForAz": false,

```

```

    "MapPublicIpOnLaunch": false,
    "State": "available",
    "SubnetId": "subnet-0736441d38EXAMPLE",
    "VpcId": "vpc-081ec835f3EXAMPLE",
    "OwnerId": "123456789012",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [
      {
        "AssociationId": "subnet-cidr-assoc-06c5f904499fcc623",
        "Ipv6CidrBlock": "2600:1f13:cfe:3660::/64",
        "Ipv6CidrBlockState": {
          "State": "associating"
        }
      }
    ],
    "Tags": [
      {
        "Key": "Name",
        "Value": "my-ipv4-ipv6-subnet"
      }
    ],
    "SubnetArn": "arn:aws:ec2:us-west-2:123456789012:subnet/
subnet-0736441d38EXAMPLE"
  }
}

```

Beispiel 3: Um ein Subnetz nur mit einem IPv6 CIDR Block zu erstellen

Das folgende `create-subnet` Beispiel erstellt ein Subnetz im angegebenen Bereich VPC mit dem angegebenen IPv6 CIDR Block.

```

aws ec2 create-subnet \
  --vpc-id vpc-081ec835f3EXAMPLE \
  --ipv6-native \
  --ipv6-cidr-block 2600:1f16:115:200::/64 \
  --tag-specifications ResourceType=subnet,Tags=[{Key=Name,Value=my-ipv6-only-
subnet}]

```

Ausgabe:

```

{
  "Subnet": {
    "AvailabilityZone": "us-west-2a",

```

```

    "AvailabilityZoneId": "usw2-az2",
    "AvailableIpAddressCount": 0,
    "DefaultForAz": false,
    "MapPublicIpOnLaunch": false,
    "State": "available",
    "SubnetId": "subnet-03f720e7deEXAMPLE",
    "VpcId": "vpc-081ec835f3EXAMPLE",
    "OwnerId": "123456789012",
    "AssignIpv6AddressOnCreation": true,
    "Ipv6CidrBlockAssociationSet": [
      {
        "AssociationId": "subnet-cidr-assoc-01ef639edde556709",
        "Ipv6CidrBlock": "2600:1f13:cfe:3660::/64",
        "Ipv6CidrBlockState": {
          "State": "associating"
        }
      }
    ],
    "Tags": [
      {
        "Key": "Name",
        "Value": "my-ipv6-only-subnet"
      }
    ],
    "SubnetArn": "arn:aws:ec2:us-west-2:123456789012:subnet/
subnet-03f720e7deEXAMPLE"
  }
}

```

Weitere Informationen finden Sie unter [VPCs und Subnetze](#) im VPC Amazon-Benutzerhandbuch.

- API-Einheiten finden Sie [CreateSubnet](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Subnetz mit den angegebenen CIDR Werten erstellt.

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

Ausgabe:

```

AvailabilityZone      : us-west-2c
AvailableIpAddressCount : 251
CidrBlock             : 10.0.0.0/24
DefaultForAz         : False
MapPublicIpOnLaunch  : False
State                 : pending
SubnetId              : subnet-1a2b3c4d
Tag                   : {}
VpcId                 : vpc-12345678

```

- API-Einheiten finden Sie unter [CreateSubnet AWS Tools for PowerShell](#) Cmdlet-Referenz.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

require 'aws-sdk-ec2'

# Creates a subnet within a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the subnet.
#
# Prerequisites:
#
# - A VPC in Amazon VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the subnet.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param availability_zone [String] The ID of the Availability Zone
#   for the subnet.
# @param tag_key [String] The key portion of the tag for the subnet.
# @param tag_value [String] The value portion of the tag for the subnet.

```

```
# @return [Boolean] true if the subnet was created and tagged;
# otherwise, false.
# @example
# exit 1 unless subnet_created_and_tagged?(
#   Aws::EC2::Resource.new(region: 'us-west-2'),
#   'vpc-6713dfEX',
#   '10.0.0.0/24',
#   'us-west-2a',
#   'my-key',
#   'my-value'
# )
def subnet_created_and_tagged?(
  ec2_resource,
  vpc_id,
  cidr_block,
  availability_zone,
  tag_key,
  tag_value
)
  subnet = ec2_resource.create_subnet(
    vpc_id: vpc_id,
    cidr_block: cidr_block,
    availability_zone: availability_zone
  )
  subnet.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts "Subnet created with ID '#{subnet.id}' in VPC with ID '#{vpc_id}' " \
    "and CIDR block '#{cidr_block}' in availability zone " \
    "'#{availability_zone}' and tagged with key '#{tag_key}' and " \
    "value '#{tag_value}'."
  true
rescue StandardError => e
  puts "Error creating or tagging subnet: #{e.message}"
  false
end

# Example usage:
def run_me
```



```
vpc_id = ''
cidr_block = ''
availability_zone = ''
tag_key = ''
tag_value = ''
region = ''
# Print usage information and then stop.
if ARGV[0] == '--help' || ARGV[0] == '-h'
  puts 'Usage:  ruby ec2-ruby-example-create-subnet.rb ' \
    'VPC_ID CIDR_BLOCK AVAILABILITY_ZONE TAG_KEY TAG_VALUE REGION'
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  puts 'Example: ruby ec2-ruby-example-create-subnet.rb ' \
    'vpc-6713dfEX 10.0.0.0/24 us-west-2a my-key my-value us-west-2'
  exit 1
# If no values are specified at the command prompt, use these default values.
elsif ARGV.count.zero?
  vpc_id = 'vpc-6713dfEX'
  cidr_block = '10.0.0.0/24'
  availability_zone = 'us-west-2a'
  tag_key = 'my-key'
  tag_value = 'my-value'
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  region = 'us-west-2'
# Otherwise, use the values as specified at the command prompt.
else
  vpc_id = ARGV[0]
  cidr_block = ARGV[1]
  availability_zone = ARGV[2]
  tag_key = ARGV[3]
  tag_value = ARGV[4]
  region = ARGV[5]
end

ec2_resource = Aws::EC2::Resource.new(region: region)

if subnet_created_and_tagged?(
  ec2_resource,
  vpc_id,
  cidr_block,
  availability_zone,
  tag_key,
  tag_value
)
  puts 'Subnet created and tagged.'
```

```

else
  puts 'Subnet not created or not tagged.'
end
end

run_me if $PROGRAM_NAME == __FILE__

```

- API-Einheiten finden Sie [CreateSubnet](#) unter AWS SDK for Ruby API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie es **CreateTags** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `CreateTags`.

C++

SDK für C++

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

//! Add or overwrite only the specified tags for the specified Amazon Elastic
Compute Cloud (Amazon EC2) resource or resources.
/*!
  \param resources: The resources for the tags.
  \param tags: Vector of tags.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::createTags(const Aws::Vector<Aws::String> &resources,
                             const Aws::Vector<Aws::EC2::Model::Tag> &tags,
                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

```

```

Aws::EC2::Model::CreateTagsRequest createTagsRequest;
createTagsRequest.SetResources(resources);
createTagsRequest.SetTags(tags);

Aws::EC2::Model::CreateTagsOutcome outcome =
ec2Client.CreateTags(createTagsRequest);

if (outcome.IsSuccess()) {
    std::cout << "Successfully created tags for resources" << std::endl;
} else {
    std::cerr << "Failed to create tags for resources, " <<
outcome.GetError().GetMessage() << std::endl;
}

return outcome.IsSuccess();
}

```

- API-Einheiten finden Sie [CreateTags](#) unter AWS SDK for C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: Um einer Ressource ein Tag hinzuzufügen

Das folgende `create-tags` Beispiel fügt das Tag dem angegebenen Bild `Stack=production` hinzu oder überschreibt ein vorhandenes Tag für den AMI Ort, an dem sich der Tag-Schlüssel befindet `Stack`.

```

aws ec2 create-tags \
  --resources ami-1234567890abcdef0 \
  --tags Key=Stack,Value=production

```

Weitere Informationen finden Sie unter [Dies ist der Thementitel](#) im Amazon Elastic Compute Cloud-Benutzerhandbuch für Linux-Instances.

Beispiel 2: So fügen Sie mehreren Ressourcen Tags hinzu

Im folgenden `create-tags` Beispiel werden zwei Tags für eine Instanz AMI und eine Instanz hinzugefügt (oder überschrieben). Eines der Tags hat einen Schlüssel (`webserver`), aber

keinen Wert (Wert ist auf eine leere Zeichenfolge festgelegt). Das andere Tag hat einen Schlüssel (`stack`) und einen Wert (`Production`).

```
aws ec2 create-tags \
  --resources ami-1a2b3c4d i-1234567890abcdef0 \
  --tags Key=webserver,Value= Key=stack,Value=Production
```

Weitere Informationen finden Sie unter [Dies ist der Thementitel](#) im Amazon Elastic Compute Cloud-Benutzerhandbuch für Linux-Instances.

Beispiel 3: Um Tags hinzuzufügen, die Sonderzeichen enthalten

Das folgende `create-tags`-Beispiel fügt das Tag `[Group]=test` für eine Instance hinzu. Die eckigen Klammern (`[` und `]`) sind Sonderzeichen und müssen mit Escape-Zeichen versehen werden. In den folgenden Beispielen wird auch das Zeilenfortsetzungszeichen verwendet, das für jede Umgebung geeignet ist.

Wenn Sie Windows verwenden, schließen Sie das Element, das Sonderzeichen enthält, in doppelte Anführungszeichen (`„`) ein und stellen Sie jedem doppelten Anführungszeichen wie folgt einen umgekehrten Schrägstrich (`\`) voran:

```
aws ec2 create-tags ^
  --resources i-1234567890abcdef0 ^
  --tags Key=\"[Group]\",Value=test
```

Wenn Sie Windows verwenden PowerShell, setzen Sie für das Element den Wert, der Sonderzeichen enthält, doppelte Anführungszeichen (`„`), stellen Sie jedem doppelten Anführungszeichen einen umgekehrten Schrägstrich (`\`) voran und setzen Sie dann die gesamte Schlüssel- und Wertstruktur wie folgt in einfache Anführungszeichen (`'`):

```
aws ec2 create-tags `
  --resources i-1234567890abcdef0 `
  --tags 'Key=\"[Group]\",Value=test'
```

Wenn Sie Linux oder OS X verwenden, schließen Sie das Element mit den Sonderzeichen mit doppelten Anführungszeichen (`„`) ein und dann die gesamte Schlüssel- und Wertstruktur mit einfachen Anführungszeichen (`'`) wie folgt:

```
aws ec2 create-tags \
```

```
--resources i-1234567890abcdef0 \  
--tags 'Key="[Group]",Value=test'
```

Weitere Informationen finden Sie unter [Dies ist der Thementitel](#) im Amazon Elastic Compute Cloud-Benutzerhandbuch für Linux-Instances.

- API-Einheiten finden Sie [CreateTags](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebenen Ressource ein einzelnes Tag hinzugefügt. Der Tag-Schlüssel ist 'myTag' und der Tag-Wert ist 'myTagValue'. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

Beispiel 2: In diesem Beispiel werden die angegebenen Tags der angegebenen Ressource aktualisiert oder hinzugefügt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },  
    @{ Key="test"; Value="anotherTagValue" } )
```

Beispiel 3: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um das Tag für den Tag-Parameter zu erstellen.

```
$tag = New-Object Amazon.EC2.Model.Tag  
$tag.Key = "myTag"  
$tag.Value = "myTagValue"  
  
New-EC2Tag -Resource i-12345678 -Tag $tag
```

- API-Einheiten finden Sie unter [CreateTags AWS Tools for PowerShell](#) Cmdlet-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

In diesem Beispiel wird das Name-Tag nach dem Erstellen einer Instanz angewendet.

```
pub async fn create_instance<'a>(
    &self,
    image_id: &'a str,
    instance_type: InstanceType,
    key_pair: &'a KeyPairInfo,
    security_groups: Vec<&'a SecurityGroup>,
) -> Result<String, EC2Error> {
    let run_instances = self
        .client
        .run_instances()
        .image_id(image_id)
        .instance_type(instance_type)
        .key_name(
            key_pair
                .key_name()
                .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?,
        )
        .set_security_group_ids(Some(
            security_groups
                .iter()
                .filter_map(|sg| sg.group_id.clone())
                .collect(),
        ))
        .min_count(1)
        .max_count(1)
        .send()
        .await?;

    if run_instances.instances().is_empty() {
        return Err(EC2Error::new("Failed to create instance"));
    }
}
```

```
    }

    let instance_id = run_instances.instances()[0].instance_id().unwrap();
    let response = self
        .client
        .create_tags()
        .resources(instance_id)
        .tags(
            Tag::builder()
                .key("Name")
                .value("From SDK Examples")
                .build(),
        )
        .send()
        .await;

    match response {
        Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
        Err(err) => {
            tracing::info!("Error applying tags to {instance_id}: {err:?}");
            return Err(err.into());
        }
    }

    tracing::info!("Instance is created.");

    Ok(instance_id.to_string())
}
```

- API-Einheiten finden Sie [CreateTags](#) in der AWS SDK API Rust-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **CreateVolume** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CreateVolume`.

CLI

AWS CLI

Um ein leeres General Purpose SSD (GP2) -Volume zu erstellen

Im folgenden `create-volume` Beispiel wird ein 80-GiB-Volume für allgemeine Zwecke SSD (GP2) in der angegebenen Availability Zone erstellt. Beachten Sie, dass die aktuelle Region angegeben werden muss `us-east-1`, oder Sie können den `--region` Parameter hinzufügen, um die Region für den Befehl anzugeben.

```
aws ec2 create-volume \  
  --volume-type gp2 \  
  --size 80 \  
  --availability-zone us-east-1a
```

Ausgabe:

```
{  
  "AvailabilityZone": "us-east-1a",  
  "Tags": [],  
  "Encrypted": false,  
  "VolumeType": "gp2",  
  "VolumeId": "vol-1234567890abcdef0",  
  "State": "creating",  
  "Iops": 240,  
  "SnapshotId": "",  
  "CreateTime": "YYYY-MM-DDTHH:MM:SS.000Z",  
  "Size": 80  
}
```

Wenn Sie keinen Volumetyp angeben, ist der Standard-Volumetyp `gp2`.

```
aws ec2 create-volume \  
  --size 80 \  
  --availability-zone us-east-1a
```

Beispiel 2: So erstellen Sie ein bereitgestelltes Volume IOPS SSD (`io1`) aus einem Snapshot

Im folgenden `create-volume` Beispiel wird mithilfe des angegebenen Snapshots ein bereitgestelltes Volume IOPS SSD (`io1`) erstellt, auf dem 1000 IOPS in der angegebenen Availability Zone bereitgestellt werden.


```
aws ec2 create-volume \  
  --volume-type io1 \  
  --iops 1000 \  
  --snapshot-id snap-066877671789bd71b \  
  --availability-zone us-east-1a
```

Ausgabe:

```
{  
  "AvailabilityZone": "us-east-1a",  
  "Tags": [],  
  "Encrypted": false,  
  "VolumeType": "io1",  
  "VolumeId": "vol-1234567890abcdef0",  
  "State": "creating",  
  "Iops": 1000,  
  "SnapshotId": "snap-066877671789bd71b",  
  "CreateTime": "YYYY-MM-DDTHH:MM:SS.000Z",  
  "Size": 500  
}
```

Beispiel 3: So erstellen Sie ein verschlüsseltes Volume

Im folgenden `create-volume` Beispiel wird ein verschlüsseltes Volume unter Verwendung der CMK EBS Standardverschlüsselung erstellt. Wenn die Verschlüsselung standardmäßig deaktiviert ist, müssen Sie den `--encrypted` Parameter wie folgt angeben.

```
aws ec2 create-volume \  
  --size 80 \  
  --encrypted \  
  --availability-zone us-east-1a
```

Ausgabe:

```
{  
  "AvailabilityZone": "us-east-1a",  
  "Tags": [],  
  "Encrypted": true,  
  "VolumeType": "gp2",  
  "VolumeId": "vol-1234567890abcdef0",  
  "State": "creating",  
}
```

```
"Iops": 240,  
"SnapshotId": "",  
"CreateTime": "YYYY-MM-DDTHH:MM:SS.000Z",  
"Size": 80  
}
```

Wenn die Standardverschlüsselung aktiviert ist, erstellt der folgende Beispielbefehl auch ohne den `--encrypted` Parameter ein verschlüsseltes Volume.

```
aws ec2 create-volume \  
  --size 80 \  
  --availability-zone us-east-1a
```

Wenn Sie den `--kms-key-id` Parameter verwenden, um ein vom Kunden verwaltetes Objekt anzugeben CMK, müssen Sie den `--encrypted` Parameter auch dann angeben, wenn die Verschlüsselung standardmäßig aktiviert ist.

```
aws ec2 create-volume \  
  --volume-type gp2 \  
  --size 80 \  
  --encrypted \  
  --kms-key-id 0ea3fef3-80a7-4778-9d8c-1c0c6EXAMPLE \  
  --availability-zone us-east-1a
```

Beispiel 4: Um ein Volume mit Tags zu erstellen

Das folgende `create-volume` Beispiel erstellt ein Volumen und fügt zwei Tags hinzu.

```
aws ec2 create-volume \  
  --availability-zone us-east-1a \  
  --volume-type gp2 \  
  --size 80 \  
  --tag-specifications  
  'ResourceType=volume,Tags=[{Key=purpose,Value=production},{Key=cost-center,Value=cc123}]'
```

- API-Einheiten finden Sie [CreateVolume](#) unter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Volumen erstellt.

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

Ausgabe:

```
Attachments      : {}
AvailabilityZone  : us-west-2a
CreateTime       : 12/22/2015 1:42:07 AM
Encrypted        : False
Iops             : 150
KmsKeyId         :
Size            : 50
SnapshotId      :
State           : creating
Tags            : {}
VolumeId        : vol-12345678
VolumeType      : gp2
```

Beispiel 2: Diese Beispielanforderung erstellt ein Volume und wendet ein Tag mit einem Stack-Schlüssel und einem Produktionswert an.

```
$tag = @{ Key="stack"; Value="production" }

$tagspec = new-object Amazon.EC2.Model.TagSpecification
$tagspec.ResourceType = "volume"
$tagspec.Tags.Add($tag)

New-EC2Volume -Size 80 -AvailabilityZone "us-west-2a" -TagSpecification $tagspec
```

- API-Einheiten finden Sie unter [CreateVolume AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie es **CreateVpc** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `CreateVpc`.

CLI

AWS CLI

Beispiel 1: Um ein zu erstellen VPC

Im folgenden `create-vpc` Beispiel wird ein VPC mit dem angegebenen IPv4 CIDR Block und einem Namen-Tag erstellt.

```
aws ec2 create-vpc \  
  --cidr-block 10.0.0.0/16 \  
  --tag-specifications ResourceType=vpc,Tags=[{Key=Name,Value=MyVpc}]
```

Ausgabe:

```
{  
  "Vpc": {  
    "CidrBlock": "10.0.0.0/16",  
    "DhcpOptionsId": "dopt-5EXAMPLE",  
    "State": "pending",  
    "VpcId": "vpc-0a60eb65b4EXAMPLE",  
    "OwnerId": "123456789012",  
    "InstanceTenancy": "default",  
    "Ipv6CidrBlockAssociationSet": [],  
    "CidrBlockAssociationSet": [  
      {  
        "AssociationId": "vpc-cidr-assoc-07501b79ecEXAMPLE",  
        "CidrBlock": "10.0.0.0/16",  
        "CidrBlockState": {  
          "State": "associated"  
        }  
      }  
    ],  
    "IsDefault": false,  
    "Tags": [  
      {  
        "Key": "Name",  
        "Value": "MyVpc"  
      }  
    ]  
  }  
}
```

```

    ]
  }
}

```

Beispiel 2: So erstellen Sie eine VPC mit einem dedizierten Mandantenverhältnis

Im folgenden `create-vpc` Beispiel wird ein Mandantenverhältnis VPC mit dem angegebenen IPv4 CIDR Block und einem dedizierten Tenancy erstellt.

```

aws ec2 create-vpc \
  --cidr-block 10.0.0.0/16 \
  --instance-tenancy dedicated

```

Ausgabe:

```

{
  "Vpc": {
    "CidrBlock": "10.0.0.0/16",
    "DhcpOptionsId": "dopt-19edf471",
    "State": "pending",
    "VpcId": "vpc-0a53287fa4EXAMPLE",
    "OwnerId": "111122223333",
    "InstanceTenancy": "dedicated",
    "Ipv6CidrBlockAssociationSet": [],
    "CidrBlockAssociationSet": [
      {
        "AssociationId": "vpc-cidr-assoc-00b24cc1c2EXAMPLE",
        "CidrBlock": "10.0.0.0/16",
        "CidrBlockState": {
          "State": "associated"
        }
      }
    ],
    "IsDefault": false
  }
}

```

Beispiel 3: Um eine VPC mit einem IPv6 CIDR Block zu erstellen

Im folgenden `create-vpc` Beispiel wird ein Block VPC mit einem von Amazon bereitgestellten Block IPv6 CIDR erstellt.

```
aws ec2 create-vpc \  
  --cidr-block 10.0.0.0/16 \  
  --amazon-provided-ipv6-cidr-block
```

Ausgabe:

```
{  
  "Vpc": {  
    "CidrBlock": "10.0.0.0/16",  
    "DhcpOptionsId": "dopt-dEXAMPLE",  
    "State": "pending",  
    "VpcId": "vpc-0fc5e3406bEXAMPLE",  
    "OwnerId": "123456789012",  
    "InstanceTenancy": "default",  
    "Ipv6CidrBlockAssociationSet": [  
      {  
        "AssociationId": "vpc-cidr-assoc-068432c60bEXAMPLE",  
        "Ipv6CidrBlock": "",  
        "Ipv6CidrBlockState": {  
          "State": "associating"  
        },  
        "Ipv6Pool": "Amazon",  
        "NetworkBorderGroup": "us-west-2"  
      }  
    ],  
    "CidrBlockAssociationSet": [  
      {  
        "AssociationId": "vpc-cidr-assoc-0669f8f9f5EXAMPLE",  
        "CidrBlock": "10.0.0.0/16",  
        "CidrBlockState": {  
          "State": "associated"  
        }  
      }  
    ],  
    "IsDefault": false  
  }  
}
```

Beispiel 4: Um einen VPC mit einem CIDR aus einem Pool zu erstellen IPAM

Im folgenden `create-vpc` Beispiel wird ein Pool VPC mit einem CIDR aus einem Amazon VPC IP Address Manager (IPAM) -Pool erstellt.

Linux und macOS:

```
aws ec2 create-vpc \  
  --ipv4-ipam-pool-id ipam-pool-0533048da7d823723 \  
  --tag-specifications  
  ResourceType=vpc,Tags=' [{Key=Environment,Value="Preprod"}  
{Key=Owner,Value="Build Team"}]'
```

Windows:

```
aws ec2 create-vpc ^  
  --ipv4-ipam-pool-id ipam-pool-0533048da7d823723 ^  
  --tag-specifications  
  ResourceType=vpc,Tags=[{Key=Environment,Value="Preprod"}, {Key=Owner,Value="Build  
Team"}]
```

Ausgabe:

```
{  
  "Vpc": {  
    "CidrBlock": "10.0.1.0/24",  
    "DhcpOptionsId": "dopt-2afccf50",  
    "State": "pending",  
    "VpcId": "vpc-010e1791024eb0af9",  
    "OwnerId": "123456789012",  
    "InstanceTenancy": "default",  
    "Ipv6CidrBlockAssociationSet": [],  
    "CidrBlockAssociationSet": [  
      {  
        "AssociationId": "vpc-cidr-assoc-0a77de1d803226d4b",  
        "CidrBlock": "10.0.1.0/24",  
        "CidrBlockState": {  
          "State": "associated"  
        }  
      }  
    ],  
    "IsDefault": false,  
    "Tags": [  
      {  
        "Key": "Environment",  
        "Value": "Preprod"  
      }  
    ],  
  },  
}
```

```
        {
            "Key": "Owner",
            "Value": "Build Team"
        }
    ]
}
}
```

Weitere Informationen finden Sie unter [Erstellen eines VPC, das einen IPAM Pool verwendet](#) [CIDR](#) im VPC IPAM Amazon-Benutzerhandbuch.

- API Einzelheiten finden Sie [CreateVpc](#) in der AWS CLI Befehlsreferenz.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * @param string $cidr
 * @return array
 */
public function createVpc(string $cidr): array
{
    try {
        $result = $this->ec2Client->createVpc([
            "CidrBlock" => $cidr,
        ]);
        return $result['Vpc'];
    } catch (Ec2Exception $caught) {
        echo "There was a problem creating the VPC: {$caught->getAwsErrorMessage()}\n";
        throw $caught;
    }
}
```


- API-Einheiten finden Sie [CreateVpc](#) unter AWS SDK for PHP API-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein VPC mit dem angegebenen erstellt CIDR. Amazon erstellt VPC außerdem Folgendes für VPC: einen DHCP Standardoptionssatz, eine Hauptrouteentabelle und ein StandardnetzwerkACL.

```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

Ausgabe:

```
CidrBlock      : 10.0.0.0/16
DhcpOptionsId  : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault      : False
State          : pending
Tags           : {}
VpcId          : vpc-12345678
```

- API-Einheiten finden Sie unter [CreateVpc AWS Tools for PowerShell](#) Cmdlet-Referenz.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
require 'aws-sdk-ec2'

# Creates a virtual private cloud (VPC) in
```

```
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param tag_key [String] The key portion of the tag for the VPC.
# @param tag_value [String] The value portion of the tag for the VPC.
# @return [Boolean] true if the VPC was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless vpc_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     '10.0.0.0/24',
#     'my-key',
#     'my-value'
#   )
def vpc_created_and_tagged?(
  ec2_resource,
  cidr_block,
  tag_key,
  tag_value
)
  vpc = ec2_resource.create_vpc(cidr_block: cidr_block)

  # Create a public DNS by enabling DNS support and DNS hostnames.
  vpc.modify_attribute(enable_dns_support: { value: true })
  vpc.modify_attribute(enable_dns_hostnames: { value: true })

  vpc.create_tags(tags: [{ key: tag_key, value: tag_value }])

  puts "Created VPC with ID '#{vpc.id}' and tagged with key " \
    "'#{tag_key}' and value '#{tag_value}'."
  true
rescue StandardError => e
  puts e.message
  false
end

# Example usage:
def run_me
  cidr_block = ''
  tag_key = ''
  tag_value = ''
```

```
region = ''
# Print usage information and then stop.
if ARGV[0] == '--help' || ARGV[0] == '-h'
  puts 'Usage:  ruby ec2-ruby-example-create-vpc.rb ' \
    'CIDR_BLOCK TAG_KEY TAG_VALUE REGION'
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  puts 'Example: ruby ec2-ruby-example-create-vpc.rb ' \
    '10.0.0.0/24 my-key my-value us-west-2'
  exit 1
# If no values are specified at the command prompt, use these default values.
elsif ARGV.count.zero?
  cidr_block = '10.0.0.0/24'
  tag_key = 'my-key'
  tag_value = 'my-value'
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  region = 'us-west-2'
# Otherwise, use the values as specified at the command prompt.
else
  cidr_block = ARGV[0]
  tag_key = ARGV[1]
  tag_value = ARGV[2]
  region = ARGV[3]
end

ec2_resource = Aws::EC2::Resource.new(region: region)

if vpc_created_and_tagged?(
  ec2_resource,
  cidr_block,
  tag_key,
  tag_value
)
  puts 'VPC created and tagged.'
else
  puts 'VPC not created or not tagged.'
end
end

run_me if $PROGRAM_NAME == __FILE__
```

- API-Einheiten finden Sie [CreateVpc](#) unter AWS SDK for Ruby API-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie es **CreateVpcEndpoint** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `CreateVpcEndpoint`.

CLI

AWS CLI

Beispiel 1: Um einen Gateway-Endpunkt zu erstellen

Das folgende `create-vpc-endpoint` Beispiel erstellt einen VPC Gateway-Endpunkt zwischen VPC `vpc-1a2b3c4d` und Amazon S3 in der `us-east-1` Region und verknüpft die Routentabelle `rtb-11aa22bb` mit dem Endpunkt.

```
aws ec2 create-vpc-endpoint \  
  --vpc-id vpc-1a2b3c4d \  
  --service-name com.amazonaws.us-east-1.s3 \  
  --route-table-ids rtb-11aa22bb
```

Ausgabe:

```
{  
  "VpcEndpoint": {  
    "PolicyDocument": "{\"Version\":\"2008-10-17\",\"Statement\":[{\"Sid\":\"\",\"Effect\":\"Allow\",\"Principal\":\"*\",\"Action\":\"*\",\"Resource\":\"*\"}]}\",  
    "VpcId": "vpc-1a2b3c4d",  
    "State": "available",  
    "ServiceName": "com.amazonaws.us-east-1.s3",  
    "RouteTableIds": [  
      "rtb-11aa22bb"  
    ],  
    "VpcEndpointId": "vpc-1a2b3c4d",  
    "CreationTimestamp": "2015-05-15T09:40:50Z"  
  }  
}
```

Weitere Informationen finden Sie im AWS PrivateLink Handbuch unter [Erstellen eines Gateway-Endpunkts](#).

Beispiel 2: So erstellen Sie einen Schnittstellen-Endpunkt

Das folgende `create-vpc-endpoint` Beispiel erstellt einen VPC Schnittstellenendpunkt zwischen VPC `vpc-1a2b3c4d` und Amazon S3 in der `us-east-1` Region. Der Befehl erstellt den Endpunkt im Subnetz `subnet-1a2b3c4d`, ordnet ihn einer Sicherheitsgruppe `sg-1a2b3c4d` zu und fügt ein Tag mit dem Schlüssel „Service“ und dem Wert „S3“ hinzu.

```
aws ec2 create-vpc-endpoint \  
  --vpc-id vpc-1a2b3c4d \  
  --vpc-endpoint-type Interface \  
  --service-name com.amazonaws.us-east-1.s3 \  
  --subnet-ids subnet-7b16de0c \  
  --security-group-id sg-1a2b3c4d \  
  --tag-specifications ResourceType=vpc-endpoint,Tags=[{Key=service,Value=S3}]
```

Ausgabe:

```
{  
  "VpcEndpoint": {  
    "VpcEndpointId": "vpce-1a2b3c4d5e6f1a2b3",  
    "VpcEndpointType": "Interface",  
    "VpcId": "vpc-1a2b3c4d",  
    "ServiceName": "com.amazonaws.us-east-1.s3",  
    "State": "pending",  
    "RouteTableIds": [],  
    "SubnetIds": [  
      "subnet-1a2b3c4d"  
    ],  
    "Groups": [  
      {  
        "GroupId": "sg-1a2b3c4d",  
        "GroupName": "default"  
      }  
    ],  
    "PrivateDnsEnabled": false,  
    "RequesterManaged": false,  
    "NetworkInterfaceIds": [  
      "eni-0b16f0581c8ac6877"  
    ],  
  },  
}
```

```

    "DnsEntries": [
      {
        "DnsName": "*.vpce-1a2b3c4d5e6f1a2b3-9hnenorg.s3.us-
east-1.vpce.amazonaws.com",
        "HostedZoneId": "Z7HUB22UULQXV"
      },
      {
        "DnsName": "*.vpce-1a2b3c4d5e6f1a2b3-9hnenorg-us-east-1c.s3.us-
east-1.vpce.amazonaws.com",
        "HostedZoneId": "Z7HUB22UULQXV"
      }
    ],
    "CreationTimestamp": "2021-03-05T14:46:16.030000+00:00",
    "Tags": [
      {
        "Key": "service",
        "Value": "S3"
      }
    ],
    "OwnerId": "123456789012"
  }
}

```

Weitere Informationen finden Sie unter [Erstellen eines Schnittstellenendpunkts](#) im Benutzerhandbuch für AWS PrivateLink

Beispiel 3: So erstellen Sie einen Gateway Load Balancer Balancer-Endpunkt

Im folgenden `create-vpc-endpoint` Beispiel wird ein Gateway Load Balancer-Endpunkt zwischen VPC `vpc-111122223333aabbcc` und einem Dienst erstellt, der mit einem Gateway Load Balancer konfiguriert ist.

```

aws ec2 create-vpc-endpoint \
  --service-name com.amazonaws.vpce.us-east-1.vpce-svc-123123a1c43abc123 \
  --vpc-endpoint-type GatewayLoadBalancer \
  --vpc-id vpc-111122223333aabbcc \
  --subnet-ids subnet-0011aabbcc2233445

```

Ausgabe:

```

{
  "VpcEndpoint": {
    "VpcEndpointId": "vpce-aabbaabbaabbaabba",

```

```

    "VpcEndpointType": "GatewayLoadBalancer",
    "VpcId": "vpc-111122223333aabbcc",
    "ServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-123123a1c43abc123",
    "State": "pending",
    "SubnetIds": [
        "subnet-0011aabbcc2233445"
    ],
    "RequesterManaged": false,
    "NetworkInterfaceIds": [
        "eni-01010120203030405"
    ],
    "CreationTimestamp": "2020-11-11T08:06:03.522Z",
    "OwnerId": "123456789012"
}
}

```

Weitere Informationen finden Sie unter [Gateway Load Balancer-Endpoints](#) im Benutzerhandbuch für AWS PrivateLink

- API-Einheiten finden Sie [CreateVpcEndpoint](#) in der AWS CLI Befehlsreferenz.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

/**
 * @param string $serviceName
 * @param string $vpcId
 * @param array $routeTableIds
 * @return array
 */
public function createVpcEndpoint(string $serviceName, string $vpcId, array
$routeTableIds): array
{
    try {

```

```

        $result = $this->ec2Client->createVpcEndpoint([
            'ServiceName' => $serviceName,
            'VpcId' => $vpcId,
            'RouteTableIds' => $routeTableIds,
        ]);

        return $result["VpcEndpoint"];
    } catch (Ec2Exception $caught) {
        echo "There was a problem creating the VPC Endpoint: {"$caught->getAwsErrorMessage()}\n";
        throw $caught;
    }
}

```

- API-Einheiten finden Sie [CreateVpcEndpoint](#) unter AWS SDK for PHP API-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein neuer VPC-Endpunkt für den Service `com.amazonaws.eu-west-1.s3` im `vpc-0fc1ff23f45b678eb` erstellt.

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
vpc-0fc1ff23f45b678eb
```

Ausgabe:

```
ClientToken VpcEndpoint
-----
                Amazon.EC2.Model.VpcEndpoint
```

- API-Einheiten [CreateVpcEndpoint AWS Tools for PowerShell](#) finden Sie unter Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie `CreateVpnConnection` mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CreateVpnConnection`.

CLI

AWS CLI

Beispiel 1: Um eine VPN Verbindung mit dynamischem Routing herzustellen

Im folgenden `create-vpn-connection` Beispiel wird eine VPN Verbindung zwischen dem angegebenen Virtual Private Gateway und dem angegebenen Kunden-Gateway hergestellt und Tags auf die VPN Verbindung angewendet. Die Ausgabe enthält die Konfigurationsinformationen für Ihr Kunden-Gateway-Gerät im XML Format.

```
aws ec2 create-vpn-connection \
  --type ipsec.1 \
  --customer-gateway-id cgw-001122334455aabbc \
  --vpn-gateway-id vgw-1a1a1a1a1a1a2b2b2 \
  --tag-specification 'ResourceType=vpn-connection,Tags=[{Key=Name,Value=BGP-VPN}]'
```

Ausgabe:

```
{
  "VpnConnection": {
    "CustomerGatewayConfiguration": "...configuration information...",
    "CustomerGatewayId": "cgw-001122334455aabbc",
    "Category": "VPN",
    "State": "pending",
    "VpnConnectionId": "vpn-123123123123abcab",
    "VpnGatewayId": "vgw-1a1a1a1a1a1a2b2b2",
    "Options": {
      "EnableAcceleration": false,
      "StaticRoutesOnly": false,
      "LocalIpv4NetworkCidr": "0.0.0.0/0",
      "RemoteIpv4NetworkCidr": "0.0.0.0/0",
      "TunnelInsideIpVersion": "ipv4",
      "TunnelOptions": [
        {},
        {}
      ]
    }
  },
}
```

```

    "Routes": [],
    "Tags": [
      {
        "Key": "Name",
        "Value": "BGP-VPN"
      }
    ]
  }
}

```

Weitere Informationen finden Sie im AWS Site-to-Site VPN Benutzerhandbuch unter [So AWS Site-to-Site VPN funktioniert](#) es.

Beispiel 2: Um eine VPN Verbindung mit statischem Routing herzustellen

Im folgenden `create-vpn-connection` Beispiel wird eine VPN Verbindung zwischen dem angegebenen Virtual Private Gateway und dem angegebenen Kunden-Gateway hergestellt. Die Optionen spezifizieren statisches Routing. Die Ausgabe enthält die Konfigurationsinformationen für Ihr Kunden-Gateway-Gerät im XML Format.

```

aws ec2 create-vpn-connection \
  --type ipsec.1 \
  --customer-gateway-id cgw-001122334455aabbc \
  --vpn-gateway-id vgw-1a1a1a1a1a1a2b2b2 \
  --options "{\"StaticRoutesOnly\":true}"

```

Ausgabe:

```

{
  "VpnConnection": {
    "CustomerGatewayConfiguration": "..configuration information...",
    "CustomerGatewayId": "cgw-001122334455aabbc",
    "Category": "VPN",
    "State": "pending",
    "VpnConnectionId": "vpn-123123123123abcab",
    "VpnGatewayId": "vgw-1a1a1a1a1a1a2b2b2",
    "Options": {
      "EnableAcceleration": false,
      "StaticRoutesOnly": true,
      "LocalIpv4NetworkCidr": "0.0.0.0/0",
      "RemoteIpv4NetworkCidr": "0.0.0.0/0",
      "TunnelInsideIpVersion": "ipv4",
    }
  }
}

```

```

        "TunnelOptions": [
            {},
            {}
        ]
    },
    "Routes": [],
    "Tags": []
}
}

```

Weitere Informationen finden Sie im AWS Site-to-Site VPN Benutzerhandbuch unter [So AWS Site-to-Site VPN funktioniert](#) es.

Beispiel 3: So stellen Sie eine VPN Verbindung her und geben Ihren eigenen internen CIDR und vorab gemeinsam genutzten Schlüssel an

Im folgenden `create-vpn-connection` Beispiel wird eine VPN Verbindung hergestellt und der interne CIDR IP-Adressblock sowie ein benutzerdefinierter Pre-Shared-Schlüssel für jeden Tunnel angegeben. Die angegebenen Werte werden in den `CustomerGatewayConfiguration` Informationen zurückgegeben.

```

aws ec2 create-vpn-connection \
  --type ipsec.1 \
  --customer-gateway-id cgw-001122334455aabb \
  --vpn-gateway-id vgw-1a1a1a1a1a1a2b2b \
  --options
  TunnelOptions='[{TunnelInsideCidr=169.254.12.0/30,PreSharedKey=ExamplePreSharedKey1},
  {TunnelInsideCidr=169.254.13.0/30,PreSharedKey=ExamplePreSharedKey2}]'

```

Ausgabe:

```

{
  "VpnConnection": {
    "CustomerGatewayConfiguration": "..configuration information...",
    "CustomerGatewayId": "cgw-001122334455aabb",
    "Category": "VPN",
    "State": "pending",
    "VpnConnectionId": "vpn-123123123123abcab",
    "VpnGatewayId": "vgw-1a1a1a1a1a1a2b2b",
    "Options": {
      "EnableAcceleration": false,
      "StaticRoutesOnly": false,
    }
  }
}

```

```

    "LocalIpv4NetworkCidr": "0.0.0.0/0",
    "RemoteIpv4NetworkCidr": "0.0.0.0/0",
    "TunnelInsideIpVersion": "ipv4",
    "TunnelOptions": [
      {
        "OutsideIpAddress": "203.0.113.3",
        "TunnelInsideCidr": "169.254.12.0/30",
        "PreSharedKey": "ExamplePreSharedKey1"
      },
      {
        "OutsideIpAddress": "203.0.113.5",
        "TunnelInsideCidr": "169.254.13.0/30",
        "PreSharedKey": "ExamplePreSharedKey2"
      }
    ]
  },
  "Routes": [],
  "Tags": []
}
}

```

Weitere Informationen finden Sie im AWS Site-to-Site VPN Benutzerhandbuch unter [So AWS Site-to-Site VPN funktioniert](#) es.

Beispiel 4: So erstellen Sie eine VPN Verbindung, die IPv6 Traffic unterstützt

Im folgenden `create-vpn-connection` Beispiel wird eine VPN Verbindung erstellt, die den IPv6 Verkehr zwischen dem angegebenen Transit-Gateway und dem angegebenen Kunden-Gateway unterstützt. Die Tunneloptionen für beide Tunnel geben an, dass die IKE Verhandlung eingeleitet AWS werden muss.

```

aws ec2 create-vpn-connection \
  --type ipsec.1 \
  --transit-gateway-id tgw-12312312312312312 \
  --customer-gateway-id cgw-001122334455aabbcc \
  --options TunnelInsideIpVersion=ipv6,TunnelOptions=[{StartupAction=start},
{StartupAction=start}]

```

Ausgabe:

```

{
  "VpnConnection": {

```

```

"CustomerGatewayConfiguration": "..configuration information...",
"CustomerGatewayId": "cgw-001122334455aabbcc",
"Category": "VPN",
"State": "pending",
"VpnConnectionId": "vpn-11111111122222222",
"TransitGatewayId": "tgw-12312312312312312",
"Options": {
  "EnableAcceleration": false,
  "StaticRoutesOnly": false,
  "LocalIpv6NetworkCidr": "::/0",
  "RemoteIpv6NetworkCidr": "::/0",
  "TunnelInsideIpVersion": "ipv6",
  "TunnelOptions": [
    {
      "OutsideIpAddress": "203.0.113.3",
      "StartupAction": "start"
    },
    {
      "OutsideIpAddress": "203.0.113.5",
      "StartupAction": "start"
    }
  ]
},
"Routes": [],
"Tags": []
}
}

```

Weitere Informationen finden Sie im AWS Site-to-Site VPN Benutzerhandbuch unter [So AWS Site-to-Site VPN funktioniert](#) es.

- API Einheiten finden Sie [CreateVpnConnection](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine VPN Verbindung zwischen dem angegebenen Virtual Private Gateway und dem angegebenen Kunden-Gateway hergestellt. Die Ausgabe enthält die Konfigurationsinformationen, die Ihr Netzwerkadministrator benötigt, im XML Format.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId vgw-1a2b3c4d
```

Ausgabe:

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                       :
Routes                       : {}
State                        : pending
Tags                         : {}
Type                         :
VgwTelemetry                 : {}
VpnConnectionId             : vpn-12345678
VpnGatewayId                 : vgw-1a2b3c4d
```

Beispiel 2: In diesem Beispiel wird die VPN Verbindung hergestellt und die Konfiguration in einer Datei mit dem angegebenen Namen erfasst.

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-configuration.xml
```

Beispiel 3: In diesem Beispiel wird eine VPN Verbindung mit statischem Routing zwischen dem angegebenen virtuellen privaten Gateway und dem angegebenen Kunden-Gateway hergestellt.

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- API-Einheiten finden Sie unter [CreateVpnConnection AWS Tools for PowerShell Cmdlet](#)-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **CreateVpnConnectionRoute** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CreateVpnConnectionRoute`.

CLI

AWS CLI

Um eine statische Route für eine VPN Verbindung zu erstellen

In diesem Beispiel wird eine statische Route für die angegebene VPN Verbindung erstellt. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 create-vpn-connection-route --vpn-connection-id vpn-40f41529 --  
destination-cidr-block 11.12.0.0/16
```

- API Einzelheiten finden Sie [CreateVpnConnectionRoute](#) unter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene statische Route für die angegebene VPN Verbindung erstellt.

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock  
11.12.0.0/16
```

- API Einzelheiten finden Sie unter [CreateVpnConnectionRoute AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter. [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **CreateVpnGateway** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `CreateVpnGateway`.

CLI

AWS CLI

Um ein virtuelles privates Gateway zu erstellen

In diesem Beispiel wird ein virtuelles privates Gateway erstellt.

Befehl:

```
aws ec2 create-vpn-gateway --type ipsec.1
```

Ausgabe:

```
{
  "VpnGateway": {
    "AmazonSideAsn": 64512,
    "State": "available",
    "Type": "ipsec.1",
    "VpnGatewayId": "vgw-9a4cacf3",
    "VpcAttachments": []
  }
}
```

Um ein virtuelles privates Gateway mit einer bestimmten Amazon-Seite zu erstellen ASN

In diesem Beispiel wird ein virtuelles privates Gateway erstellt und die Autonomous System Number (ASN) für die Amazon-Seite der BGP Sitzung angegeben.

Befehl:

```
aws ec2 create-vpn-gateway --type ipsec.1 --amazon-side-asn 65001
```

Ausgabe:

```
{
  "VpnGateway": {
    "AmazonSideAsn": 65001,
    "State": "available",
    "Type": "ipsec.1",
  }
}
```



```
    "VpnGatewayId": "vgw-9a4cacf3",  
    "VpcAttachments": []  
  }  
}
```

- API-Einheiten finden Sie [CreateVpnGateway](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene virtuelle private Gateway erstellt.

```
New-EC2VpnGateway -Type ipsec.1
```

Ausgabe:

```
AvailabilityZone :  
State           : available  
Tags            : {}  
Type            : ipsec.1  
VpcAttachments  : {}  
VpnGatewayId    : vgw-1a2b3c4d
```

- API-Einheiten finden Sie unter [CreateVpnGateway AWS Tools for PowerShell Cmdlet](#)-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DeleteCustomerGateway** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DeleteCustomerGateway`.

CLI

AWS CLI

Um ein Kunden-Gateway zu löschen

In diesem Beispiel wird das angegebene Kunden-Gateway gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-customer-gateway --customer-gateway-id cgw-0e11f167
```

- API-Einheiten finden Sie [DeleteCustomerGateway](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Kunden-Gateway gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2CustomerGateway (DeleteCustomerGateway)" on
Target "cgw-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- API-Einheiten finden Sie unter [DeleteCustomerGateway AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DeleteDhcpOptions** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DeleteDhcpOptions`.

CLI

AWS CLI

Um einen DHCP Optionssatz zu löschen

In diesem Beispiel wird der angegebene DHCP Optionssatz gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-dhcp-options --dhcp-options-id dopt-d9070ebb
```

- API-Einheiten finden Sie [DeleteDhcpOptions](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene DHCP Optionssatz gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2DhcpOption (DeleteDhcpOptions)" on Target
"dopt-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- API-Einheiten finden Sie unter [DeleteDhcpOptions AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DeleteFlowLogs** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DeleteFlowLogs`.

CLI

AWS CLI

Um ein Flow-Protokoll zu löschen

Im folgenden `delete-flow-logs` Beispiel wird das angegebene Flow-Protokoll gelöscht.

```
aws ec2 delete-flow-logs --flow-log-id fl-11223344556677889
```

Ausgabe:

```
{  
  "Unsuccessful": []  
}
```

- API-Einheiten finden Sie [DeleteFlowLogs](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel entfernt den angegebenen Wert `FlowLogId fl-01a2b3456a789c01`

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

Ausgabe:

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-EC2FlowLog (DeleteFlowLogs)" on target  
"fl-01a2b3456a789c01".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

- API-Einheiten finden [DeleteFlowLogs AWS Tools for PowerShell](#) Sie unter Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie `DeleteInternetGateway` mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DeleteInternetGateway`.

CLI

AWS CLI

Um ein Internet-Gateway zu löschen

Im folgenden `delete-internet-gateway` Beispiel wird das angegebene Internet-Gateway gelöscht.

```
aws ec2 delete-internet-gateway \  
  --internet-gateway-id igw-0d0fb496b3EXAMPLE
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Internet-Gateways](#) im VPCAmazon-Benutzerhandbuch.

- APIEinzelheiten finden Sie [DeleteInternetGateway](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Internet-Gateway gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den `Force`-Parameter angeben.

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Ausgabe:

```
Confirm  
Are you sure you want to perform this action?
```

```
Performing operation "Remove-EC2InternetGateway (DeleteInternetGateway)" on
Target "igw-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- API-Einheiten finden Sie unter [DeleteInternetGateway AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie es **DeleteKeyPair** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DeleteKeyPair`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Delete an Amazon EC2 key pair.
/// </summary>
/// <param name="keyPairName">The name of the key pair to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteKeyPair(string keyPairName)
{
    try
```

```
    {
        await _amazonEC2.DeleteKeyPairAsync(new
DeleteKeyPairRequest(keyPairName)).ConfigureAwait(false);
        return true;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
        {
            _logger.LogError($"KeyPair {keyPairName} does not exist and
cannot be deleted. Please verify the key pair name and try again.");
        }

        return false;
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Couldn't delete the key pair because:
{ex.Message}");
        return false;
    }
}

/// <summary>
/// Delete the temporary file where the key pair information was saved.
/// </summary>
/// <param name="tempFileName">The path to the temporary file.</param>
public void DeleteTempFile(string tempFileName)
{
    if (File.Exists(tempFileName))
    {
        File.Delete(tempFileName);
    }
}
}
```

- API Einheiten finden Sie [DeleteKeyPair](#) unter AWS SDK for .NET API Referenz.

Bash

AWS CLI mit Bash-Skript

 Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#####
# function ec2_delete_keypair
#
# This function deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair.
#
# Parameters:
#     -n key_pair_name - A key pair name.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_delete_keypair() {
    local key_pair_name response

    local option OPTARG # Required to use getopt command in a function.
    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_delete_keypair"
        echo "Deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair."
        echo "  -n key_pair_name - A key pair name."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "n:h" option; do
        case "${option}" in
            n) key_pair_name="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
        esac
    done
}
```



```

    \?)
    echo "Invalid parameter"
    usage
    return 1
    ;;
esac
done
export OPTIND=1

if [[ -z "$key_pair_name" ]]; then
    errecho "ERROR: You must provide a key pair name with the -n parameter."
    usage
    return 1
fi

response=$(aws ec2 delete-key-pair \
  --key-name "$key_pair_name") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-key-pair operation failed.$response"
    return 1
}

return 0
}

```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:

```

```

#      $1 - The error code returned by the AWS CLI.
#
# Returns:
#      0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then
        errecho " The service returned an error."
    elif [ "$err_code" == 255 ]; then
        errecho " 255 is a catch-all error."
    fi

    return 0
}

```

- API-Einheiten finden Sie [DeleteKeyPair](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#!/ Delete an Amazon Elastic Compute Cloud (Amazon EC2) instance key pair.
```

```

/ * !
  \ param keyPairName: A name for a key pair.
  \ param clientConfiguration: AWS client configuration.
  \ return bool: Function succeeded.
 * /

bool AwsDoc::EC2::deleteKeyPair(const Aws::String &keyPairName,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DeleteKeyPairRequest request;

    request.SetKeyName(keyPairName);
    const Aws::EC2::Model::DeleteKeyPairOutcome outcome =
ec2Client.DeleteKeyPair(
    request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete key pair " << keyPairName <<
            ":" << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted key pair named " << keyPairName <<
            std::endl;
    }

    return outcome.IsSuccess();
}

```

- API-Einheiten finden Sie [DeleteKeyPair](#) unter AWS SDK for C++ API-Referenz.

CLI

AWS CLI

So löschen Sie ein Schlüsselpaar

Im folgenden `delete-key-pair` Beispiel wird das angegebene key pair gelöscht.

```
aws ec2 delete-key-pair \
  --key-name my-key-pair
```

Ausgabe:

```
{
  "Return": true,
  "KeyPairId": "key-03c8d3aceb53b507"
}
```

Weitere Informationen finden Sie unter [Erstellen und Löschen von Schlüsselpaaren](#) im Benutzerhandbuch für die AWS Befehlszeilenschnittstelle.

- API-Einheiten finden Sie [DeleteKeyPair](#) unter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Deletes a key pair asynchronously.
 *
 * @param keyPair the name of the key pair to delete
 * @return a {@link CompletableFuture} that represents the result of the
 * asynchronous operation.
 *       The {@link CompletableFuture} will complete with a {@link
 * DeleteKeyPairResponse} object
 *       that provides the result of the key pair deletion operation.
 */
public CompletableFuture<DeleteKeyPairResponse> deleteKeysAsync(String
keyPair) {
    DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
        .keyName(keyPair)
        .build();

    // Initiate the asynchronous request to delete the key pair.
    CompletableFuture<DeleteKeyPairResponse> response =
getAsyncClient().deleteKeyPair(request);
    return response.whenComplete((resp, ex) -> {
        if (ex != null) {
```

```

        throw new RuntimeException("Failed to delete key pair: " +
keyPair, ex);
    } else if (resp == null) {
        throw new RuntimeException("No response received for deleting key
pair: " + keyPair);
    }
    });
}

```

- API-Einheiten finden Sie [DeleteKeyPair](#) unter AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import { DeleteKeyPairCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Deletes the specified key pair, by removing the public key from Amazon EC2.
 * @param {{ keyName: string }} options
 */
export const main = async ({ keyName }) => {
  const client = new EC2Client({});
  const command = new DeleteKeyPairCommand({
    KeyName: keyName,
  });

  try {
    await client.send(command);
    console.log("Successfully deleted key pair.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide the required value`);
    } else {
      throw caught;
    }
  }
}

```

```
    }  
  }  
};
```

- API-Einheiten finden Sie [DeleteKeyPair](#) unter AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
suspend fun deleteKeys(keyPair: String?) {  
    val request =  
        DeleteKeyPairRequest {  
            keyName = keyPair  
        }  
  
    Ec2Client { region = "us-west-2" }.use { ec2 ->  
        ec2.deleteKeyPair(request)  
        println("Successfully deleted key pair named $keyPair")  
    }  
}
```

- API-Einheiten finden Sie [DeleteKeyPair](#) in der AWS SDK API Kotlin-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene key pair gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2KeyPair -KeyName my-key-pair
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2KeyPair (DeleteKeyPair)" on Target "my-key-pair".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- API-Einheiten finden Sie unter [DeleteKeyPair AWS Tools for PowerShell](#) Cmdlet-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
        directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
```

```

        access to AWS EC2 services.
    :param key_file_dir: The folder where the private key information is
    stored.
        This should be a secure folder.
    :param key_pair: A dictionary representing the Boto3 KeyPair object.
        This is a high-level object that wraps key pair actions.
Optional.
    """
    self.ec2_client = ec2_client
    self.key_pair = key_pair
    self.key_file_path: Optional[str] = None
    self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2
client
and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())

    def delete(self, key_name: str) -> bool:
        """
        Deletes a key pair by its name.

        :param key_name: The name of the key pair to delete.
        :return: A boolean indicating whether the deletion was successful.
        :raises ClientError: If there is an error in deleting the key pair, for
example,
            if the key pair does not exist.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=key_name)
            logger.info(f"Successfully deleted key pair: {key_name}")
            self.key_pair = None
            return True
        except self.ec2_client.exceptions.ClientError as err:
            logger.error(f"Deletion failed for key pair: {key_name}")
            error_code = err.response["Error"]["Code"]

```



```

        if error_code == "InvalidKeyPair.NotFound":
            logger.error(
                f"The key pair '{key_name}' does not exist and cannot be
deleted. "
                "Please verify the key pair name and try again."
            )
        raise

```

- API-Einheiten finden Sie unter [DeleteKeyPair](#) Python (Boto3) API -Referenz. AWS SDK

Rust

SDK für Rust

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Wrapper für `delete_key`, der auch den privaten Backing-Schlüssel entfernt. PEM

```

pub async fn delete(self, ec2: &EC2, util: &Util) -> Result<(), EC2Error> {
    if let Some(key_pair_id) = self.key_pair.key_pair_id() {
        ec2.delete_key_pair(key_pair_id).await?;
        if let Some(key_path) = self.key_file_path() {
            if let Err(err) = util.remove(key_path) {
                eprintln!("Failed to remove {key_path:?} ({err:?})");
            }
        }
    }
    Ok(())
}

```

```

pub async fn delete_key_pair(&self, key_pair_id: &str) -> Result<(),
EC2Error> {
    let key_pair_id: String = key_pair_id.into();
    tracing::info!("Deleting key pair {key_pair_id}");
}

```

```

        self.client
            .delete_key_pair()
            .key_pair_id(key_pair_id)
            .send()
            .await?;
    Ok(())
}

```

- API-Einheiten finden Sie [DeleteKeyPair](#) in AWS SDK der Rust-Referenz. API

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

TRY.
    lo_ec2->deletekeypair( iv_keyname = iv_key_name ).
    MESSAGE 'Amazon EC2 key pair deleted.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- API-Einheiten finden Sie [DeleteKeyPair](#) unter AWS SDKSAPABAPAPI als Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie es **DeleteLaunchTemplate** mit einem AWS SDK oder CLI


Die folgenden Codebeispiele zeigen, wie man es benutzt `DeleteLaunchTemplate`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erstellen und Verwalten eines ausfallsicheren Services](#)

.NET

AWS SDK for .NET

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Delete a launch template by name.
/// </summary>
/// <param name="templateName">The name of the template to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteTemplateByName(string templateName)
{
    try
    {
        await _amazonEc2.DeleteLaunchTemplateAsync(
            new DeleteLaunchTemplateRequest()
            {
                LaunchTemplateName = templateName
            });
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode ==
            "InvalidLaunchTemplateName.NotFoundException")
        {
            _logger.LogError(
                $"Could not delete the template, the name
                {_launchTemplateName} was not found.");
        }

        throw;
    }
}
```

```
    }
    catch (Exception ex)
    {
        _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
        throw;
    }
}
```

- API-Einheiten finden Sie [DeleteLaunchTemplate](#) unter AWS SDK for .NET API-Referenz.

CLI

AWS CLI

So löschen Sie eine Startvorlage

In diesem Beispiel wird die angegebene Startvorlage gelöscht.

Befehl:

```
aws ec2 delete-launch-template --launch-template-id lt-0abcd290751193123
```

Ausgabe:

```
{
  "LaunchTemplate": {
    "LatestVersionNumber": 2,
    "LaunchTemplateId": "lt-0abcd290751193123",
    "LaunchTemplateName": "TestTemplate",
    "DefaultVersionNumber": 2,
    "CreatedBy": "arn:aws:iam::123456789012:root",
    "CreateTime": "2017-11-23T16:46:25.000Z"
  }
}
```

- API-Einheiten finden Sie [DeleteLaunchTemplate](#) unter AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
await client.send(  
  new DeleteLaunchTemplateCommand({  
    LaunchTemplateName: NAMES.launchTemplateName,  
  }),  
);
```

- API Einzelheiten finden Sie [DeleteLaunchTemplate](#) unter AWS SDK for JavaScript API Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class AutoScalingWrapper:  
    """  
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.  
    """  
  
    def __init__(  
        self,  
        resource_prefix: str,  
        inst_type: str,  
        ami_param: str,
```

```
    autoscaling_client: boto3.client,
    ec2_client: boto3.client,
    ssm_client: boto3.client,
    iam_client: boto3.client,
):
    """
    Initializes the AutoScaler class with the necessary parameters.

    :param resource_prefix: The prefix for naming AWS resources that are
    created by this class.
    :param inst_type: The type of EC2 instance to create, such as t3.micro.
    :param ami_param: The Systems Manager parameter used to look up the AMI
    that is created.
    :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
    :param ec2_client: A Boto3 EC2 client.
    :param ssm_client: A Boto3 Systems Manager client.
    :param iam_client: A Boto3 IAM client.
    """
    self.inst_type = inst_type
    self.ami_param = ami_param
    self.autoscaling_client = autoscaling_client
    self.ec2_client = ec2_client
    self.ssm_client = ssm_client
    self.iam_client = iam_client
    sts_client = boto3.client("sts")
    self.account_id = sts_client.get_caller_identity()["Account"]

    self.key_pair_name = f"{resource_prefix}-key-pair"
    self.launch_template_name = f"{resource_prefix}-template-"
    self.group_name = f"{resource_prefix}-group"

    # Happy path
    self.instance_policy_name = f"{resource_prefix}-pol"
    self.instance_role_name = f"{resource_prefix}-role"
    self.instance_profile_name = f"{resource_prefix}-prof"

    # Failure mode
    self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
    self.bad_creds_role_name = f"{resource_prefix}-bc-role"
    self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

def delete_template(self):
    """
```

```
Deletes a launch template.
"""
try:
    self.ec2_client.delete_launch_template(
        LaunchTemplateName=self.launch_template_name
    )
    self.delete_instance_profile(
        self.instance_profile_name, self.instance_role_name
    )
    log.info("Launch template %s deleted.", self.launch_template_name)
except ClientError as err:
    if (
        err.response["Error"]["Code"]
        == "InvalidLaunchTemplateName.NotFoundException"
    ):
        log.info(
            "Launch template %s does not exist, nothing to do.",
            self.launch_template_name,
        )
    log.error(f"Full error:\n\t{err}")
```

- API-Einheiten finden Sie unter [DeleteLaunchTemplate](#) Python (Boto3) API -Referenz. AWS SDK

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeleteNetworkAcl** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DeleteNetworkAcl`.

CLI

AWS CLI

Um ein Netzwerk zu löschen ACL

In diesem Beispiel wird das angegebene Netzwerk ACL gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-network-acl --network-acl-id acl-5fb85d36
```

- API-Einheiten finden Sie [DeleteNetworkAcl](#) in der AWS CLI Befehlsreferenz.

PowerShell**Tools für PowerShell**

Beispiel 1: In diesem Beispiel wird das angegebene Netzwerk ACL gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAcl (DeleteNetworkAcl)" on Target
"acl-12345678".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"):
```

- API-Einheiten finden Sie unter [DeleteNetworkAcl AWS Tools for PowerShell Cmdlet](#)-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DeleteNetworkAclEntry** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DeleteNetworkAclEntry`.

CLI

AWS CLI

Um einen ACL Netzwerkeintrag zu löschen

In diesem Beispiel wird die Eingangsregel Nummer 100 aus dem angegebenen Netzwerk gelöscht. ACL Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-number 100
```

- APIEinzelheiten finden Sie unter [DeleteNetworkAclEntry AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Regel aus dem angegebenen Netzwerk entferntACL. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkAclEntry (DeleteNetworkAclEntry)" on
Target "acl-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- APIEinzelheiten finden Sie unter [DeleteNetworkAclEntry AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DeleteNetworkInterface** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DeleteNetworkInterface`.

CLI

AWS CLI

Um eine Netzwerkschnittstelle zu löschen

In diesem Beispiel wird die angegebene Netzwerkschnittstelle gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-network-interface --network-interface-id eni-e5aa89a3
```

- API-Einheiten finden Sie [DeleteNetworkInterface](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2NetworkInterface (DeleteNetworkInterface)" on
Target "eni-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- API-Einheiten finden Sie unter [DeleteNetworkInterface AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeletePlacementGroup** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DeletePlacementGroup`.

CLI

AWS CLI

Um eine Platzierungsgruppe zu löschen

Dieser Beispielbefehl löscht die angegebene Platzierungsgruppe.

Befehl:

```
aws ec2 delete-placement-group --group-name my-cluster
```

- API-Einheiten finden Sie unter [DeletePlacementGroup AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Platzierungsgruppe gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den `Force`-Parameter angeben.

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2PlacementGroup (DeletePlacementGroup)" on Target
"my-placement-group".
```

```
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):
```

- API-Einheiten finden Sie unter [DeletePlacementGroup AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DeleteRoute** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DeleteRoute`.

CLI

AWS CLI

Um eine Route zu löschen

In diesem Beispiel wird die angegebene Route aus der angegebenen Routentabelle gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-route --route-table-id rtb-22574640 --destination-cidr-block 0.0.0.0/0
```

- API-Einheiten finden Sie unter [DeleteRoute AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Route aus der angegebenen Routentabelle gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

Ausgabe:

Confirm

Are you sure you want to perform this action?

Performing operation "Remove-EC2Route (DeleteRoute)" on Target "rtb-1a2b3c4d".

[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):

- API-Einheiten finden Sie unter [DeleteRoute AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DeleteRouteTable** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DeleteRouteTable`.

CLI

AWS CLI

Um eine Routentabelle zu löschen

In diesem Beispiel wird die angegebene Routentabelle gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-route-table --route-table-id rtb-22574640
```

- API-Einheiten finden Sie [DeleteRouteTable](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Routentabelle gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2RouteTable (DeleteRouteTable)" on Target
"rtb-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- API-Einheiten finden Sie unter [DeleteRouteTable AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie es **DeleteSecurityGroup** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DeleteSecurityGroup`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Delete an Amazon EC2 security group.
/// </summary>
/// <param name="groupName">The name of the group to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
```

```

public async Task<bool> DeleteSecurityGroup(string groupId)
{
    try
    {
        var response =
            await _amazonEC2.DeleteSecurityGroupAsync(
                new DeleteSecurityGroupRequest { GroupId = groupId });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
        {
            _logger.LogError(
                $"Security Group {groupId} does not exist and cannot be
deleted. Please verify the ID and try again.");
        }

        return false;
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
        return false;
    }
}

```

- API-Einheiten finden Sie [DeleteSecurityGroup](#) unter AWS SDK for .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#####
```

```
# function ec2_delete_security_group
#
# This function deletes an Amazon Elastic Compute Cloud (Amazon EC2) security
# group.
#
# Parameters:
#     -i security_group_id - The ID of the security group to delete.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_delete_security_group() {
    local security_group_id response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_delete_security_group"
        echo "Deletes an Amazon Elastic Compute Cloud (Amazon EC2) security group."
        echo "  -i security_group_id - The ID of the security group to delete."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "i:h" option; do
        case "${option}" in
            i) security_group_id="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1

    if [[ -z "$security_group_id" ]]; then
        errecho "ERROR: You must provide a security group ID with the -i parameter."
        usage
    fi
}
```



```

    return 1
fi

response=$(aws ec2 delete-security-group --group-id "$security_group_id" --
output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports delete-security-group operation failed.$response"
    return 1
}

return 0
}

```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then

```

```

    errecho " Command line failed to parse."
elif [ "$err_code" == 130 ]; then
    errecho " Process received SIGINT."
elif [ "$err_code" == 252 ]; then
    errecho " Command syntax invalid."
elif [ "$err_code" == 253 ]; then
    errecho " The system environment or configuration was invalid."
elif [ "$err_code" == 254 ]; then
    errecho " The service returned an error."
elif [ "$err_code" == 255 ]; then
    errecho " 255 is a catch-all error."
fi

return 0
}

```

- API-Einheiten finden Sie [DeleteSecurityGroup](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

//! Delete a security group.
/!*
 \param securityGroupID: A security group ID.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::EC2::deleteSecurityGroup(const Aws::String &securityGroupID,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DeleteSecurityGroupRequest request;

    request.SetGroupId(securityGroupID);

```

```
Aws::EC2::Model::DeleteSecurityGroupOutcome outcome =
ec2Client.DeleteSecurityGroup(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to delete security group " << securityGroupID <<
        ":" << outcome.GetError().GetMessage() << std::endl;
} else {
    std::cout << "Successfully deleted security group " << securityGroupID <<
        std::endl;
}

return outcome.IsSuccess();
}
```

- API-Einheiten finden Sie [DeleteSecurityGroup](#) unter AWS SDK for C++ API-Referenz.

CLI

AWS CLI

[EC2-Classic] Um eine Sicherheitsgruppe zu löschen

In diesem Beispiel wird die Sicherheitsgruppe mit dem Namen `MySecurityGroup` gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-security-group --group-name MySecurityGroup
```

[EC2-VPC] Um eine Sicherheitsgruppe zu löschen

In diesem Beispiel wird die Sicherheitsgruppe mit der ID `sg-903004f8` gelöscht. Beachten Sie, dass Sie für EC2 - VPC nicht anhand des Namens auf eine Sicherheitsgruppe verweisen können. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-security-group --group-id sg-903004f8
```

Weitere Informationen finden Sie unter [Verwenden von Sicherheitsgruppen](#) im Benutzerhandbuch für die AWS -Befehlszeilenschnittstelle.

- API-Einheiten finden Sie [DeleteSecurityGroup](#) unter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Deletes an EC2 security group asynchronously.
 *
 * @param groupId the ID of the security group to delete
 * @return a CompletableFuture that completes when the security group is
 * deleted
 */
public CompletableFuture<Void> deleteEC2SecGroupAsync(String groupId) {
    DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
        .groupId(groupId)
        .build();

    CompletableFuture<DeleteSecurityGroupResponse> response =
getAsyncClient().deleteSecurityGroup(request);
    return response.whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to delete security group with
Id " + groupId, ex);
        } else if (resp == null) {
            throw new RuntimeException("No response received for deleting
security group with Id " + groupId);
        }
    }).thenApply(resp -> null);
}
```

- API-Einheiten finden Sie [DeleteSecurityGroup](#) unter AWS SDK for Java 2.x APIReferenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import { DeleteSecurityGroupCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Deletes a security group.
 * @param {{ groupId: string }} options
 */
export const main = async ({ groupId }) => {
  const client = new EC2Client({});
  const command = new DeleteSecurityGroupCommand({
    GroupId: groupId,
  });

  try {
    await client.send(command);
    console.log("Security group deleted successfully.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
      console.warn(`${caught.message}. Please provide a valid GroupId.`);
    } else {
      throw caught;
    }
  }
};
```

- API Einzelheiten finden Sie [DeleteSecurityGroup](#) unter AWS SDK for JavaScript API Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
suspend fun deleteEC2SecGroup(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.deleteSecurityGroup(request)
        println("Successfully deleted Security Group with id $groupIdVal")
    }
}
```

- API-Einheiten finden Sie [DeleteSecurityGroup](#) in der AWS SDK API Kotlin-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Sicherheitsgruppe für EC2 - VPC gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2SecurityGroup (DeleteSecurityGroup)" on Target
"sg-12345678".
```

[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):

Beispiel 2: In diesem Beispiel wird die angegebene Sicherheitsgruppe für EC2 -Classic gelöscht.

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- API-Einheiten finden Sie unter [DeleteSecurityGroup AWS Tools for PowerShell](#) Cmdlet-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
    actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
    None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
        security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
        high-level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group
```

```
@classmethod
def from_client(cls) -> "SecurityGroupWrapper":
    """
    Creates a SecurityGroupWrapper instance with a default EC2 client.

    :return: An instance of SecurityGroupWrapper initialized with the default
    EC2 client.
    """
    ec2_client = boto3.client("ec2")
    return cls(ec2_client)

def delete(self, security_group_id: str) -> bool:
    """
    Deletes the specified security group.

    :param security_group_id: The ID of the security group to delete.
    Required.

    :returns: True if the deletion is successful.
    :raises ClientError: If the security group cannot be deleted due to an
    AWS service error.
    """
    try:
        self.ec2_client.delete_security_group(GroupId=security_group_id)
        logger.info(f"Successfully deleted security group
        '{security_group_id}'")
        return True
    except ClientError as err:
        logger.error(f"Deletion failed for security group
        '{security_group_id}'")
        error_code = err.response["Error"]["Code"]

        if error_code == "InvalidGroup.NotFound":
            logger.error(
                f"Security group '{security_group_id}' cannot be deleted
                because it does not exist."
            )
        elif error_code == "DependencyViolation":
            logger.error(
                f"Security group '{security_group_id}' cannot be deleted
                because it is still in use."
                "\n\t- Detached from resources"
```



```

        "\n\t- Removed from references in other groups"
        "\n\t- Removed from VPC's as a default group"
    )
    raise

```

- API-Einheiten finden Sie unter [DeleteSecurityGroup](#) Python (Boto3) API -Referenz. AWS SDK

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

pub async fn delete_security_group(&self, group_id: &str) -> Result<(),
EC2Error> {
    tracing::info!("Deleting security group {group_id}");
    self.client
        .delete_security_group()
        .group_id(group_id)
        .send()
        .await?;
    Ok(())
}

```

- API-Einheiten finden Sie [DeleteSecurityGroup](#) in der AWS SDK API Rust-Referenz.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
TRY.  
    lo_ec2->deletesecuritygroup( iv_groupid = iv_security_group_id ).  
    MESSAGE 'Security group deleted.' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.  
    MESSAGE lv_error TYPE 'E'.  
ENDTRY.
```

- API-Einheiten finden Sie [DeleteSecurityGroup](#) unter AWS SDKSAPABAPAPI als Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie es **DeleteSnapshot** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DeleteSnapshot`.

CLI

AWS CLI

So löschen Sie einen Snapshot

Dieser Beispielbefehl löscht einen Snapshot mit der Snapshot-ID von `snap-1234567890abcdef0`. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-snapshot --snapshot-id snap-1234567890abcdef0
```

- API-Einheiten finden Sie [DeleteSnapshot](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Snapshot gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Snapshot (DeleteSnapshot)" on target
"snap-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- API-Einheiten finden Sie unter [DeleteSnapshot AWS Tools for PowerShell](#) Cmdlet-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
async fn delete_snapshot(client: &Client, id: &str) -> Result<(), Error> {
    client.delete_snapshot().snapshot_id(id).send().await?;
```

```
println!("Deleted");  
  
Ok(())  
}
```

- API-Einheiten finden Sie [DeleteSnapshot](#) in der AWS SDK API Rust-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DeleteSpotDatafeedSubscription** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DeleteSpotDatafeedSubscription`.

CLI

AWS CLI

Um ein Spot-Instance-Datenfeed-Abonnement zu kündigen

Dieser Beispielbefehl löscht ein Spot-Datenfeed-Abonnement für das Konto. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-spot-datafeed-subscription
```

- API-Einheiten finden Sie [DeleteSpotDatafeedSubscription](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird Ihr Spot-Instance-Datenfeed gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2SpotDatafeedSubscription
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2SpotDatafeedSubscription
(DeleteSpotDatafeedSubscription)" on Target "".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- API-Einheiten finden Sie unter [DeleteSpotDatafeedSubscription AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeleteSubnet** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung von `DeleteSubnet`.

CLI

AWS CLI

Um ein Subnetz zu löschen

In diesem Beispiel wird das angegebene Subnetz gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-subnet --subnet-id subnet-9d4a7b6c
```

- API-Einheiten finden Sie unter [DeleteSubnet AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Subnetz gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2Subnet (DeleteSubnet)" on Target
"subnet-1a2b3c4d".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- API-Einheiten finden Sie unter [DeleteSubnet AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeleteTags** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung von `DeleteTags`.

CLI

AWS CLI

Beispiel 1: Um ein Tag aus einer Ressource zu löschen

Im folgenden `delete-tags` Beispiel wird das Tag `Stack=Test` aus dem angegebenen Bild gelöscht. Wenn Sie sowohl einen Wert als auch einen Schlüsselnamen angeben, wird das Tag nur gelöscht, wenn der Wert des Tags dem angegebenen Wert entspricht.

```
aws ec2 delete-tags \
  --resources ami-1234567890abcdef0 \
```

```
--tags Key=Stack, Value=Test
```

Es ist optional, den Wert für ein Tag anzugeben. Im folgenden `delete-tags` Beispiel wird das Tag mit dem Schlüsselnamen `purpose` aus der angegebenen Instanz gelöscht, unabhängig vom Tag-Wert für das Tag.

```
aws ec2 delete-tags \  
  --resources i-1234567890abcdef0 \  
  --tags Key=purpose
```

Wenn Sie die leere Zeichenfolge als Tag-Wert angeben, wird das Tag nur gelöscht, wenn der Tag-Wert eine leere Zeichenfolge ist. Im folgenden `delete-tags` Beispiel wird die leere Zeichenfolge als Tag-Wert für das zu löschende Tag angegeben.

```
aws ec2 delete-tags \  
  --resources i-1234567890abcdef0 \  
  --tags Key=Name, Value=
```

Beispiel 2: Um ein Tag aus mehreren Ressourcen zu löschen

Das folgende `delete-tags` Beispiel löscht das Tag `purpose=test` sowohl aus einer Instanz als auch aus einer AMI. Wie im vorherigen Beispiel gezeigt, können Sie den Tag-Wert im Befehl weglassen.

```
aws ec2 delete-tags \  
  --resources i-1234567890abcdef0 ami-1234567890abcdef0 \  
  --tags Key=Purpose
```

- API-Einheiten finden Sie unter [DeleteTags AWS CLI Befehlsreferenz](#).

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Tag unabhängig vom Tag-Wert aus der angegebenen Ressource gelöscht. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

Beispiel 2: In diesem Beispiel wird das angegebene Tag aus der angegebenen Ressource gelöscht, aber nur, wenn der Tag-Wert übereinstimmt. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -Force
```

Beispiel 3: In diesem Beispiel wird das angegebene Tag unabhängig vom Tag-Wert aus der angegebenen Ressource gelöscht.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

Beispiel 4: In diesem Beispiel wird das angegebene Tag aus der angegebenen Ressource gelöscht, aber nur, wenn der Tag-Wert übereinstimmt.

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- API-Einheiten finden Sie unter [DeleteTags AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DeleteVolume** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DeleteVolume`.

CLI

AWS CLI

Um ein Volume zu löschen

Dieser Beispielbefehl löscht ein verfügbares Volume mit der Volume-ID `vol-049df61146c4d7901`. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-volume --volume-id vol-049df61146c4d7901
```

- API-Einheiten finden Sie unter [DeleteVolume AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Volume getrennt. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2Volume -VolumeId vol-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Volume (DeleteVolume)" on target
"vol-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- API-Einheiten finden Sie unter [DeleteVolume AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie es **DeleteVpc** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DeleteVpc`.

CLI

AWS CLI

Um ein zu löschen VPC

In diesem Beispiel wird das angegebene VPC gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-vpc --vpc-id vpc-a01106c2
```

- API-Einheiten finden Sie [DeleteVpc](#) in der AWS CLI Befehlsreferenz.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * @param string $vpcId
 * @return void
 */
public function deleteVpc(string $vpcId)
{
    try {
        $this->ec2Client->deleteVpc([
            "VpcId" => $vpcId,
        ]);
    } catch (Ec2Exception $caught){
        echo "There was a problem deleting the VPC: {$caught-
>getAwsErrorMessage()}\n";
        throw $caught;
    }
}
```

```
}  
}
```

- API-Einheiten finden Sie [DeleteVpc](#) unter AWS SDK for PHP API-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene VPC Datei gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2Vpc -VpcId vpc-12345678
```

Ausgabe:

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-EC2Vpc (DeleteVpc)" on Target "vpc-12345678".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

- API-Einheiten finden Sie unter [DeleteVpc AWS Tools for PowerShell](#) Cmdlet-Referenz.


Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DeleteVpcEndpoint** mit einem AWS SDK

Das folgende Codebeispiel zeigt die Verwendung `DeleteVpcEndpoint`.

PHP

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * @param string $vpcEndpointId
 * @return void
 */
public function deleteVpcEndpoint(string $vpcEndpointId)
{
    try {
        $this->ec2Client->deleteVpcEndpoints([
            "VpcEndpointIds" => [$vpcEndpointId],
        ]);
    } catch (Ec2Exception $caught){
        echo "There was a problem deleting the VPC Endpoint: {$caught->getAwsErrorMessage()}\n";
        throw $caught;
    }
}
```

- API-Einheiten finden Sie [DeleteVpcEndpoint](#) unter AWS SDK for PHP API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeleteVpnConnection** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DeleteVpnConnection`.

CLI

AWS CLI

Um eine VPN Verbindung zu löschen

In diesem Beispiel wird die angegebene VPN Verbindung gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-vpn-connection --vpn-connection-id vpn-40f41529
```

- API-Einheiten finden Sie [DeleteVpnConnection](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene VPN Verbindung gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing operation "Remove-EC2VpnConnection (DeleteVpnConnection)" on Target
"vpn-12345678".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"):
```

- API-Einheiten finden Sie unter [DeleteVpnConnection AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter. [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie `DeleteVpnConnectionRoute` mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DeleteVpnConnectionRoute`.

CLI

AWS CLI

Um eine statische Route aus einer VPN Verbindung zu löschen

In diesem Beispiel wird die angegebene statische Route aus der angegebenen VPN Verbindung gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-vpn-connection-route --vpn-connection-id vpn-40f41529 --  
destination-cidr-block 11.12.0.0/16
```

- API-Einheiten finden Sie unter [DeleteVpnConnectionRoute AWS CLI Befehlsreferenz](#).

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene statische Route aus der angegebenen VPN Verbindung entfernt. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock  
11.12.0.0/16
```

Ausgabe:

```
Confirm  
Are you sure you want to perform this action?  
Performing operation "Remove-EC2VpnConnectionRoute (DeleteVpnConnectionRoute)" on  
Target "vpn-12345678".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

- API-Einheiten finden Sie unter [DeleteVpnConnectionRoute AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeleteVpnGateway** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DeleteVpnGateway`.

CLI

AWS CLI

Um ein virtuelles privates Gateway zu löschen

In diesem Beispiel wird das angegebene virtuelle private Gateway gelöscht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 delete-vpn-gateway --vpn-gateway-id vgw-9a4caf3
```

- API-Einheiten finden Sie [DeleteVpnGateway](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene virtuelle private Gateway gelöscht. Sie werden zur Bestätigung aufgefordert, bevor der Vorgang fortgesetzt wird, sofern Sie nicht auch den Force-Parameter angeben.

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Ausgabe:

```
Confirm
```

```
Are you sure you want to perform this action?  
Performing operation "Remove-EC2VpnGateway (DeleteVpnGateway)" on Target  
"vgw-1a2b3c4d".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"):
```

- API-Einheiten finden Sie unter [DeleteVpnGateway AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeregisterImage** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung von `DeregisterImage`.

CLI

AWS CLI

Um die Registrierung eines abzumelden AMI

In diesem Beispiel wird die angegebene Datei deregistriert. AMI Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 deregister-image --image-id ami-4fa54026
```

- API-Einheiten finden Sie [DeregisterImage](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Registrierung der angegebenen Daten aufgehoben. AMI

```
Unregister-EC2Image -ImageId ami-12345678
```


- API-Einheiten finden Sie unter [DeregisterImage AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeAccountAttributes** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeAccountAttributes`.

CLI

AWS CLI

Um alle Attribute für Ihr AWS Konto zu beschreiben

In diesem Beispiel werden die Attribute für Ihr AWS Konto beschrieben.

Befehl:

```
aws ec2 describe-account-attributes
```

Ausgabe:

```
{
  "AccountAttributes": [
    {
      "AttributeName": "vpc-max-security-groups-per-interface",
      "AttributeValues": [
        {
          "AttributeValue": "5"
        }
      ]
    },
    {
      "AttributeName": "max-instances",
      "AttributeValues": [
        {
          "AttributeValue": "20"
        }
      ]
    }
  ]
}
```

```
    },
    {
      "AttributeName": "supported-platforms",
      "AttributeValues": [
        {
          "AttributeValue": "EC2"
        },
        {
          "AttributeValue": "VPC"
        }
      ]
    },
    {
      "AttributeName": "default-vpc",
      "AttributeValues": [
        {
          "AttributeValue": "none"
        }
      ]
    },
    {
      "AttributeName": "max-elastic-ips",
      "AttributeValues": [
        {
          "AttributeValue": "5"
        }
      ]
    },
    {
      "AttributeName": "vpc-max-elastic-ips",
      "AttributeValues": [
        {
          "AttributeValue": "5"
        }
      ]
    }
  ]
}
```

Um ein einzelnes Attribut für Ihr AWS Konto zu beschreiben

In diesem Beispiel wird das `supported-platforms` Attribut für Ihr AWS Konto beschrieben.

Befehl:

```
aws ec2 describe-account-attributes --attribute-names supported-platforms
```

Ausgabe:

```
{
  "AccountAttributes": [
    {
      "AttributeName": "supported-platforms",
      "AttributeValues": [
        {
          "AttributeValue": "EC2"
        },
        {
          "AttributeValue": "VPC"
        }
      ]
    }
  ]
}
```

- API-Einheiten finden Sie [DescribeAccountAttributes](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird beschrieben, ob Sie Instances in EC2 -Classic und EC2 - VPC in der Region oder nur in EC2 - VPC starten können.

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

Ausgabe:

```
AttributeValue
-----
EC2
VPC
```

Beispiel 2: Dieses Beispiel beschreibt Ihre Standardeinstellung oder lautet „Keine“ VPC, wenn Sie VPC in der Region keine Standardeinstellung haben.

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

Ausgabe:

```
AttributeValue
-----
vpc-12345678
```

Beispiel 3: Dieses Beispiel beschreibt die maximale Anzahl von On-Demand-Instances, die Sie ausführen können.

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

Ausgabe:

```
AttributeValue
-----
20
```

- API-Einheiten finden Sie unter [DescribeAccountAttributes AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeAddresses** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DescribeAddresses`.

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
//! Describe all Elastic IP addresses.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::EC2::describeAddresses(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeAddressesRequest request;
    Aws::EC2::Model::DescribeAddressesOutcome outcome =
    ec2Client.DescribeAddresses(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left << std::setw(20) << "InstanceId" <<
            std::setw(15) << "Public IP" << std::setw(10) << "Domain" <<
            std::setw(30) << "Allocation ID" << std::setw(25) <<
            "NIC ID" << std::endl;

        const Aws::Vector<Aws::EC2::Model::Address> &addresses =
    outcome.GetResult().GetAddresses();
        for (const auto &address: addresses) {
            Aws::String domainString =
                Aws::EC2::Model::DomainTypeMapper::GetNameForDomainType(
                    address.GetDomain());

            std::cout << std::left << std::setw(20) <<
                address.GetInstanceId() << std::setw(15) <<
                address.GetPublicIp() << std::setw(10) << domainString <<
                std::setw(30) << address.GetAllocationId() << std::setw(25)
                << address.GetNetworkInterfaceId() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe Elastic IP addresses:" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API Einheiten finden Sie [DescribeAddresses](#) unter AWS SDK for C++ API Referenz.

CLI

AWS CLI

Beispiel 1: So rufen Sie Details über alle Ihre Elastic-IP-Adressen ab

Im folgenden `describe-addresses`-Beispiel werden Details zu Ihren Elastic-IP-Adressen angezeigt.

```
aws ec2 describe-addresses
```

Ausgabe:

```
{
  "Addresses": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "PublicIp": "198.51.100.0",
      "PublicIpv4Pool": "amazon",
      "Domain": "standard"
    },
    {
      "Domain": "vpc",
      "PublicIpv4Pool": "amazon",
      "InstanceId": "i-1234567890abcdef0",
      "NetworkInterfaceId": "eni-12345678",
      "AssociationId": "eipassoc-12345678",
      "NetworkInterfaceOwnerId": "123456789012",
      "PublicIp": "203.0.113.0",
      "AllocationId": "eipalloc-12345678",
      "PrivateIpAddress": "10.0.1.241"
    }
  ]
}
```

Beispiel 2: Um Details zu Ihren Elastic IP-Adressen abzurufen für EC2 - VPC

Im folgenden `describe-addresses` Beispiel werden Details zu Ihren Elastic IP-Adressen für die Verwendung mit Instances in a angezeigtVPC.

```
aws ec2 describe-addresses \
```

```
--filters "Name=domain,Values=vpc"
```

Ausgabe:

```
{
  "Addresses": [
    {
      "Domain": "vpc",
      "PublicIpv4Pool": "amazon",
      "InstanceId": "i-1234567890abcdef0",
      "NetworkInterfaceId": "eni-12345678",
      "AssociationId": "eipassoc-12345678",
      "NetworkInterfaceOwnerId": "123456789012",
      "PublicIp": "203.0.113.0",
      "AllocationId": "eipalloc-12345678",
      "PrivateIpAddress": "10.0.1.241"
    }
  ]
}
```

Beispiel 3: So rufen Sie Details über eine durch die Zuweisungs-ID spezifizierte Elastic-IP-Adresse ab

Im folgenden `describe-addresses` Beispiel werden Details zur Elastic IP-Adresse mit der angegebenen Zuweisungs-ID angezeigt, die einer Instance in EC2 - zugeordnet istVPC.

```
aws ec2 describe-addresses \
  --allocation-ids eipalloc-282d9641
```

Ausgabe:

```
{
  "Addresses": [
    {
      "Domain": "vpc",
      "PublicIpv4Pool": "amazon",
      "InstanceId": "i-1234567890abcdef0",
      "NetworkInterfaceId": "eni-1a2b3c4d",
      "AssociationId": "eipassoc-123abc12",
      "NetworkInterfaceOwnerId": "1234567891012",
      "PublicIp": "203.0.113.25",
      "AllocationId": "eipalloc-282d9641",
    }
  ]
}
```

```

        "PrivateIpAddress": "10.251.50.12"
      }
    ]
  }

```

Beispiel 4: Um Details zu einer Elastic IP-Adresse abzurufen, die durch ihre VPC private IP-Adresse spezifiziert ist

Im folgenden `describe-addresses` Beispiel werden Details über die Elastic IP-Adresse angezeigt, die einer bestimmten privaten IP-Adresse in EC2 - zugeordnet istVPC.

```

aws ec2 describe-addresses \
  --filters "Name=private-ip-address,Values=10.251.50.12"

```

Beispiel 5: Um Details zu Elastic IP-Adressen in EC2 -Classic abzurufen

TThelm folgenden `describe-addresses` Beispiel werden Details zu Ihren Elastic IP-Adressen für die Verwendung in EC2 -Classic angezeigt.

```

aws ec2 describe-addresses \
  --filters "Name=domain,Values=standard"

```

Ausgabe:

```

{
  "Addresses": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "PublicIp": "203.0.110.25",
      "PublicIpv4Pool": "amazon",
      "Domain": "standard"
    }
  ]
}

```

Beispiel 6: So rufen Sie Details über eine Elastic-IP-Adresse ab, die durch ihre öffentliche IP-Adresse angegeben ist

Im folgenden `describe-addresses` Beispiel werden Details zur Elastic IP-Adresse mit dem Wert angezeigt203.0.110.25, die einer Instance in EC2 -Classic zugeordnet ist.


```
aws ec2 describe-addresses \  
  --public-ips 203.0.110.25
```

Ausgabe:

```
{  
  "Addresses": [  
    {  
      "InstanceId": "i-1234567890abcdef0",  
      "PublicIp": "203.0.110.25",  
      "PublicIpv4Pool": "amazon",  
      "Domain": "standard"  
    }  
  ]  
}
```

- API-Einheiten finden Sie unter [DescribeAddresses AWS CLI-Befehlsreferenz](#).

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import { DescribeAddressesCommand, EC2Client } from "@aws-sdk/client-ec2";  
  
/**  
 * Describes the specified Elastic IP addresses or all of your Elastic IP  
 * addresses.  
 * @param {{ allocationId: string }} options  
 */  
export const main = async ({ allocationId }) => {  
  const client = new EC2Client({});  
  const command = new DescribeAddressesCommand({  
    // You can omit this property to show all addresses.  
    AllocationIds: [allocationId],  
  });
```

```
});

try {
  const { Addresses } = await client.send(command);
  const addressList = Addresses.map((address) => ` • ${address.PublicIp}`);
  console.log("Elastic IP addresses:");
  console.log(addressList.join("\n"));
} catch (caught) {
  if (
    caught instanceof Error &&
    caught.name === "InvalidAllocationID.NotFound"
  ) {
    console.warn(`${caught.message}. Please provide a valid AllocationId.`);
  } else {
    throw caught;
  }
}
};
```

- API-Einheiten finden Sie [DescribeAddresses](#) unter AWS SDK for JavaScript API-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Elastic IP-Adresse für Instances in EC2-Classic.

```
Get-EC2Address -AllocationId eipalloc-12345678
```

Ausgabe:

```
AllocationId           : eipalloc-12345678
AssociationId          : eipassoc-12345678
Domain                 : vpc
InstanceId             : i-87654321
NetworkInterfaceId    : eni-12345678
NetworkInterfaceOwnerId : 12345678
PrivateIpAddress      : 10.0.2.172
PublicIp               : 198.51.100.2
```

Beispiel 2: Dieses Beispiel beschreibt Ihre Elastic IP-Adressen für Instances in a. VPC Diese Syntax erfordert PowerShell Version 3 oder höher.

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

Beispiel 3: Dieses Beispiel beschreibt die angegebene Elastic IP-Adresse für Instances in EC2 -Classic.

```
Get-EC2Address -PublicIp 203.0.113.17
```

Ausgabe:

```
AllocationId      :  
AssociationId     :  
Domain           : standard  
InstanceId        : i-12345678  
NetworkInterfaceId :  
NetworkInterfaceOwnerId :  
PrivateIpAddress :  
PublicIp         : 203.0.113.17
```

Beispiel 4: Dieses Beispiel beschreibt Ihre Elastic IP-Adressen für Instances in EC2 -Classic. Für diese Syntax ist PowerShell Version 3 oder höher erforderlich.

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

Beispiel 5: Dieses Beispiel beschreibt all Ihre Elastic IP-Adressen.

```
Get-EC2Address
```

Beispiel 6: Dieses Beispiel gibt die öffentliche und private IP für die im Filter angegebene Instance-ID zurück

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

Ausgabe:

```
PrivateIpAddress PublicIp
```

```
-----
10.0.0.99      63.36.5.227
```

Beispiel 7: In diesem Beispiel werden alle Elastic-Daten IPs mit ihrer Zuweisungs-ID, Zuordnungs-ID und Instanz-IDs abgerufen

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId,
AllocationId, PublicIp
```

Ausgabe:

InstanceId	AssociationId	AllocationId	PublicIp
-----	-----	-----	-----
		eipalloc-012e3b456789e1fad	
17.212.120.178			
i-0c123dfd3415bac67	eipassoc-0e123456bb7890bdb	eipalloc-01cd23ebf45f7890c	
17.212.124.77			
		eipalloc-012345678eeabcfad	
17.212.225.7			
i-0123d405c67e89a0c	eipassoc-0c123b456783966ba	eipalloc-0123cdd456a8f7892	
37.216.52.173			
i-0f1bf2f34c5678d09	eipassoc-0e12934568a952d96	eipalloc-0e1c23e4d5e6789e4	
37.218.222.278			
i-012e3cb4df567e8aa	eipassoc-0d1b2fa4d67d03810	eipalloc-0123f456f78a01b58	
37.210.82.27			
i-0123bcf4b567890e1	eipassoc-01d2345f678903fb1	eipalloc-0e1db23cfef5c45c7	
37.215.222.270			

Beispiel 8: In diesem Beispiel wird eine Liste von EC2 IP-Adressen abgerufen, die dem Tag-Schlüssel 'Category' mit dem Wert 'Prod' entsprechen

```
Get-EC2Address -Filter @{Name="tag:Category";Values="Prod"}
```

Ausgabe:

```
AllocationId      : eipalloc-0123f456f81a01b58
AssociationId     : eipassoc-0d1b23a456d103810
CustomerOwnedIp  :
CustomerOwnedIpv4Pool :
```

```

Domain           : vpc
InstanceId       : i-012e3cb4df567e1aa
NetworkBorderGroup : eu-west-1
NetworkInterfaceId : eni-0123f41d5a60d5f40
NetworkInterfaceOwnerId : 123456789012
PrivateIpAddress  : 192.168.1.84
PublicIp         : 34.250.81.29
PublicIpv4Pool    : amazon
Tags             : {Category, Name}

```

- API-Einheiten finden Sie unter [DescribeAddresses AWS Tools for PowerShell](#) Cmdlet-Referenz.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

TRY.
    oo_result = lo_ec2->describeaddresses( ) .
    oo_result is returned for testing purposes. "
    DATA(lt_addresses) = oo_result->get_addresses( ).
    MESSAGE 'Retrieved information about Elastic IP addresses.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- API-Einheiten finden Sie [DescribeAddresses](#) unter AWS SDKSAPABAPAPI als Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeAvailabilityZones** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DescribeAvailabilityZones`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erstellen und Verwalten eines ausfallsicheren Services](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
/// </summary>
/// <returns>A list of availability zones.</returns>
public async Task<List<string>> DescribeAvailabilityZones()
{
    try
    {
        var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
            new DescribeAvailabilityZonesRequest());
        return zoneResponse.AvailabilityZones.Select(z =>
z.ZoneName).ToList();
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
        throw;
    }
    catch (Exception ex)
    {
```

```

        _logger.LogError($"An error occurred while listing availability
zones.: {ex.Message}");
        throw;
    }
}

```

- API-Einheiten finden Sie [DescribeAvailabilityZones](#) unter AWS SDK for .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

//! DescribeAvailabilityZones
/*!
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
int AwsDoc::EC2::describeAvailabilityZones(const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeAvailabilityZonesRequest request;
    Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
ec2Client.DescribeAvailabilityZones(request);

    if (outcome.IsSuccess()) {
        std::cout << std::left <<
            std::setw(32) << "ZoneName" <<
            std::setw(20) << "State" <<
            std::setw(32) << "Region" << std::endl;

        const auto &zones =
            outcome.GetResult().GetAvailabilityZones();

        for (const auto &zone: zones) {

```

```

        Aws::String stateString =

    Aws::EC2::Model::AvailabilityZoneStateMapper::GetNameForAvailabilityZoneState(
        zone.GetState());
        std::cout << std::left <<
            std::setw(32) << zone.GetZoneName() <<
            std::setw(20) << stateString <<
            std::setw(32) << zone.GetRegionName() << std::endl;
    }
} else {
    std::cerr << "Failed to describe availability zones:" <<
        outcome.GetError().GetMessage() << std::endl;

}

return outcome.IsSuccess();
}

```

- API-Einheiten finden Sie [DescribeAvailabilityZones](#) unter AWS SDK for C++ API-Referenz.

CLI

AWS CLI

So beschreiben Sie Ihre Availability Zones

Das folgende Beispiel `describe-availability-zones` zeigt Details zu den Availability Zones, die für Sie verfügbar sind. Die Antwort umfasst nur Availability Zones für die aktuelle Region. In diesem Beispiel wird die Standardregion `us-west-2` (Oregon) des Profils verwendet.

```
aws ec2 describe-availability-zones
```

Ausgabe:

```

{
  "AvailabilityZones": [
    {
      "State": "available",
      "OptInStatus": "opt-in-not-required",
      "Messages": [],

```



```
    "RegionName": "us-west-2",
    "ZoneName": "us-west-2a",
    "ZoneId": "usw2-az1",
    "GroupName": "us-west-2",
    "NetworkBorderGroup": "us-west-2"
  },
  {
    "State": "available",
    "OptInStatus": "opt-in-not-required",
    "Messages": [],
    "RegionName": "us-west-2",
    "ZoneName": "us-west-2b",
    "ZoneId": "usw2-az2",
    "GroupName": "us-west-2",
    "NetworkBorderGroup": "us-west-2"
  },
  {
    "State": "available",
    "OptInStatus": "opt-in-not-required",
    "Messages": [],
    "RegionName": "us-west-2",
    "ZoneName": "us-west-2c",
    "ZoneId": "usw2-az3",
    "GroupName": "us-west-2",
    "NetworkBorderGroup": "us-west-2"
  },
  {
    "State": "available",
    "OptInStatus": "opt-in-not-required",
    "Messages": [],
    "RegionName": "us-west-2",
    "ZoneName": "us-west-2d",
    "ZoneId": "usw2-az4",
    "GroupName": "us-west-2",
    "NetworkBorderGroup": "us-west-2"
  },
  {
    "State": "available",
    "OptInStatus": "opted-in",
    "Messages": [],
    "RegionName": "us-west-2",
    "ZoneName": "us-west-2-lax-1a",
    "ZoneId": "usw2-lax1-az1",
    "GroupName": "us-west-2-lax-1",
```

```

        "NetworkBorderGroup": "us-west-2-lax-1"
    }
]
}

```

- API-Einheiten finden Sie [DescribeAvailabilityZones](#) unter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Availability Zones für die aktuelle Region beschrieben, die Ihnen zur Verfügung stehen.

```
Get-EC2AvailabilityZone
```

Ausgabe:

Messages	RegionName	State	ZoneName
{}	us-west-2	available	us-west-2a
{}	us-west-2	available	us-west-2b
{}	us-west-2	available	us-west-2c

Beispiel 2: In diesem Beispiel werden alle Availability Zones beschrieben, die sich in einem beeinträchtigten Zustand befinden. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

Beispiel 3: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um den Filter zu erstellen.

```

$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"

Get-EC2AvailabilityZone -Filter $filter

```

- API-Einheiten finden Sie unter [DescribeAvailabilityZones AWS Tools for PowerShell](#) Cmdlet-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
```

```
self.inst_type = inst_type
self.ami_param = ami_param
self.autoscaling_client = autoscaling_client
self.ec2_client = ec2_client
self.ssm_client = ssm_client
self.iam_client = iam_client
sts_client = boto3.client("sts")
self.account_id = sts_client.get_caller_identity()["Account"]

self.key_pair_name = f"{resource_prefix}-key-pair"
self.launch_template_name = f"{resource_prefix}-template-"
self.group_name = f"{resource_prefix}-group"

# Happy path
self.instance_policy_name = f"{resource_prefix}-pol"
self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"

# Failure mode
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

def get_availability_zones(self) -> List[str]:
    """
    Gets a list of Availability Zones in the AWS Region of the Amazon EC2
    client.

    :return: The list of Availability Zones for the client Region.
    """
    try:
        response = self.ec2_client.describe_availability_zones()
        zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
        log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
    except ClientError as err:
        log.error("Failed to retrieve availability zones.")
        log.error(f"Full error:\n\t{err}")
    else:
        return zones
```

- API-Einheiten finden Sie unter [DescribeAvailabilityZones](#) Python (Boto3) API - Referenz. AWS SDK

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
TRY.  
    oo_result = lo_ec2->describeavailabilityzones( ).  
    " oo_result is returned for testing purposes. "  
    DATA(lt_zones) = oo_result->get_availabilityzones( ).  
    MESSAGE 'Retrieved information about Availability Zones.' TYPE 'I'.  
  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->  
>av_err_msg }|.  
    MESSAGE lv_error TYPE 'E'.  
ENDTRY.
```

- API-Einheiten finden Sie [DescribeAvailabilityZones](#) unter AWS SDK SAP ABAP API als Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeBundleTasks** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeBundleTasks`.

CLI

AWS CLI

Um Ihre Bundle-Aufgaben zu beschreiben

In diesem Beispiel werden alle Ihre Bundle-Aufgaben beschrieben.

Befehl:

```
aws ec2 describe-bundle-tasks
```

Ausgabe:

```
{
  "BundleTasks": [
    {
      "UpdateTime": "2015-09-15T13:26:54.000Z",
      "InstanceId": "i-1234567890abcdef0",
      "Storage": {
        "S3": {
          "Prefix": "winami",
          "Bucket": "bundletasks"
        }
      },
      "State": "bundling",
      "StartTime": "2015-09-15T13:24:35.000Z",
      "Progress": "3%",
      "BundleId": "bun-2a4e041c"
    }
  ]
}
```

- API-Einheiten finden Sie [DescribeBundleTasks](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Bundle-Aufgabe.

```
Get-EC2BundleTask -BundleId bun-12345678
```

Beispiel 2: Dieses Beispiel beschreibt die Bundle-Aufgaben, deren Status entweder „abgeschlossen“ oder „fehlgeschlagen“ ist.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "complete", "failed" )

Get-EC2BundleTask -Filter $filter
```

- API-Einheiten finden Sie unter [DescribeBundleTasks](#) Cmdlet-Referenz.AWS Tools for PowerShell

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeCapacityReservations** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeCapacityReservations`.

CLI

AWS CLI

Beispiel 1: Um eine oder mehrere Ihrer Kapazitätsreservierungen zu beschreiben

Im folgenden `describe-capacity-reservations` Beispiel werden Details zu all Ihren Kapazitätsreservierungen in der aktuellen AWS Region angezeigt.

```
aws ec2 describe-capacity-reservations
```

Ausgabe:

```
{
  "CapacityReservations": [
    {
      "CapacityReservationId": "cr-1234abcd56EXAMPLE ",
      "EndDateType": "unlimited",
      "AvailabilityZone": "eu-west-1a",
      "InstanceMatchCriteria": "open",
      "Tags": [],
    }
  ]
}
```

```

    "EphemeralStorage": false,
    "CreateDate": "2019-08-16T09:03:18.000Z",
    "AvailableInstanceCount": 1,
    "InstancePlatform": "Linux/UNIX",
    "TotalInstanceCount": 1,
    "State": "active",
    "Tenancy": "default",
    "EbsOptimized": true,
    "InstanceType": "a1.medium"
  },
  {
    "CapacityReservationId": "cr-abcdEXAMPLE9876ef ",
    "EndDateType": "unlimited",
    "AvailabilityZone": "eu-west-1a",
    "InstanceMatchCriteria": "open",
    "Tags": [],
    "EphemeralStorage": false,
    "CreateDate": "2019-08-07T11:34:19.000Z",
    "AvailableInstanceCount": 3,
    "InstancePlatform": "Linux/UNIX",
    "TotalInstanceCount": 3,
    "State": "cancelled",
    "Tenancy": "default",
    "EbsOptimized": true,
    "InstanceType": "m5.large"
  }
]
}

```

Beispiel 2: Um eine oder mehrere Ihrer Kapazitätsreservierungen zu beschreiben

Im folgenden `describe-capacity-reservations` Beispiel werden Details zur angegebenen Kapazitätsreservierung angezeigt.

```

aws ec2 describe-capacity-reservations \
  --capacity-reservation-ids cr-1234abcd56EXAMPLE

```

Ausgabe:

```

{
  "CapacityReservations": [
    {
      "CapacityReservationId": "cr-1234abcd56EXAMPLE",

```



```
        "EndDateType": "unlimited",
        "AvailabilityZone": "eu-west-1a",
        "InstanceMatchCriteria": "open",
        "Tags": [],
        "EphemeralStorage": false,
        "CreateDate": "2019-08-16T09:03:18.000Z",
        "AvailableInstanceCount": 1,
        "InstancePlatform": "Linux/UNIX",
        "TotalInstanceCount": 1,
        "State": "active",
        "Tenancy": "default",
        "EbsOptimized": true,
        "InstanceType": "a1.medium"
    }
]
}
```

Weitere Informationen finden Sie unter [Kapazitätsreservierung anzeigen](#) im Amazon Elastic Compute Cloud-Benutzerhandbuch für Linux-Instances.

- API-Einheiten finden Sie [DescribeCapacityReservations](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt eine oder mehrere Ihrer Kapazitätsreservierungen für die Region

```
Get-EC2CapacityReservation -Region eu-west-1
```

Ausgabe:

```
AvailabilityZone      : eu-west-1b
AvailableInstanceCount : 2
CapacityReservationId : cr-0c1f2345db6f7cdba
CreateDate            : 3/28/2019 9:29:41 AM
EbsOptimized          : True
EndDate               : 1/1/0001 12:00:00 AM
EndDateType           : unlimited
EphemeralStorage      : False
InstanceMatchCriteria : open
```

```
InstancePlatform      : Windows
InstanceType         : m4.xlarge
State                 : active
Tags                  : {}
Tenancy                : default
TotalInstanceCount   : 2
```

- API-Einheiten finden Sie unter [DescribeCapacityReservations AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeCustomerGateways** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeCustomerGateways`.

CLI

AWS CLI

Um Ihre Kunden-Gateways zu beschreiben

Dieses Beispiel beschreibt Ihre Kunden-Gateways.

Befehl:

```
aws ec2 describe-customer-gateways
```

Ausgabe:

```
{
  "CustomerGateways": [
    {
      "CustomerGatewayId": "cgw-b4dc3961",
      "IpAddress": "203.0.113.12",
      "State": "available",
      "Type": "ipsec.1",
      "BgpAsn": "65000"
    },
  ],
}
```

```
{
  "CustomerGatewayId": "cgw-0e11f167",
  "IpAddress": "12.1.2.3",
  "State": "available",
  "Type": "ipsec.1",
  "BgpAsn": "65534"
}
]
```

Um ein bestimmtes Kunden-Gateway zu beschreiben

Dieses Beispiel beschreibt das angegebene Kunden-Gateway.

Befehl:

```
aws ec2 describe-customer-gateways --customer-gateway-ids cgw-0e11f167
```

Ausgabe:

```
{
  "CustomerGateways": [
    {
      "CustomerGatewayId": "cgw-0e11f167",
      "IpAddress": "12.1.2.3",
      "State": "available",
      "Type": "ipsec.1",
      "BgpAsn": "65534"
    }
  ]
}
```

- API-Einheiten finden Sie [DescribeCustomerGateways](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das angegebene Kunden-Gateway.

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

Ausgabe:

```
BgpAsn          : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress       : 203.0.113.12
State          : available
Tags           : {}
Type           : ipsec.1
```

Beispiel 2: Dieses Beispiel beschreibt jedes Kunden-Gateway, dessen Status entweder ausstehend oder verfügbar ist.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

Beispiel 3: Dieses Beispiel beschreibt alle Ihre Kunden-Gateways.

```
Get-EC2CustomerGateway
```

- API-Einheiten finden Sie unter [DescribeCustomerGateways AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie `DescribeDhcpOptions` mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeDhcpOptions`.

CLI**AWS CLI**

Beispiel 1: Um Ihre DHCP Optionen zu beschreiben

Im folgenden `describe-dhcp-options` Beispiel werden Details zu Ihren DHCP Optionen abgerufen.

aws ec2 describe-dhcp-options

Ausgabe:

```
{
  "DhcpOptions": [
    {
      "DhcpConfigurations": [
        {
          "Key": "domain-name",
          "Values": [
            {
              "Value": "us-east-2.compute.internal"
            }
          ]
        },
        {
          "Key": "domain-name-servers",
          "Values": [
            {
              "Value": "AmazonProvidedDNS"
            }
          ]
        }
      ],
      "DhcpOptionsId": "dopt-19edf471",
      "OwnerId": "111122223333"
    },
    {
      "DhcpConfigurations": [
        {
          "Key": "domain-name",
          "Values": [
            {
              "Value": "us-east-2.compute.internal"
            }
          ]
        },
        {
          "Key": "domain-name-servers",
          "Values": [
            {
              "Value": "AmazonProvidedDNS"
            }
          ]
        }
      ]
    }
  ]
}
```

```

        }
    ]
}
],
"DhcpOptionsId": "dopt-fEXAMPLE",
"OwnerId": "111122223333"
}
]
}

```

Weitere Informationen finden Sie unter [Arbeiten mit DHCP Optionssätzen](#) im AWS VPCBenutzerhandbuch.

Beispiel 2: Um Ihre DHCP Optionen zu beschreiben und die Ausgabe zu filtern

Das folgende `describe-dhcp-options` Beispiel beschreibt Ihre DHCP Optionen und verwendet einen Filter, um nur DHCP Optionen zurückzugeben, die `example.com` für den Domain-Namensserver verfügbar sind. Das Beispiel verwendet den `--query` Parameter, um nur die Konfigurationsinformationen und die ID in der Ausgabe anzuzeigen.

```

aws ec2 describe-dhcp-options \
  --filters Name=key,Values=domain-name-servers Name=value,Values=example.com \
  --query "DhcpOptions[*].[DhcpConfigurations,DhcpOptionsId]"

```

Ausgabe:

```

[
  [
    [
      {
        "Key": "domain-name",
        "Values": [
          {
            "Value": "example.com"
          }
        ]
      },
      {
        "Key": "domain-name-servers",
        "Values": [
          {
            "Value": "172.16.16.16"
          }
        ]
      }
    ]
  ]
]

```

```

    ],
    "dopt-001122334455667ab"
  ]
]

```

Weitere Informationen finden Sie unter [Arbeiten mit DHCP Optionssätzen](#) im AWS VPCBenutzerhandbuch.

- APIEinzelheiten finden Sie [DescribeDhcpOptions](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel listet Ihre DHCP Optionssätze auf.

```
Get-EC2DhcpOption
```

Ausgabe:

DhcpConfigurations	DhcpOptionsId	Tag
-----	-----	---
{domain-name, domain-name-servers}	dopt-1a2b3c4d	{}
{domain-name, domain-name-servers}	dopt-2a3b4c5d	{}
{domain-name-servers}	dopt-3a4b5c6d	{}

Beispiel 2: In diesem Beispiel werden Konfigurationsdetails für den angegebenen DHCP Optionssatz abgerufen.

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

Ausgabe:

Key	Values
---	-----
domain-name	{abc.local}
domain-name-servers	{10.0.0.101, 10.0.0.102}

- API-Einheiten finden Sie unter [DescribeDhcpOptions AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeFlowLogs** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeFlowLogs`.

CLI

AWS CLI

Beispiel 1: Um all Ihre Flow-Logs zu beschreiben

Im folgenden `describe-flow-logs` Beispiel werden Details für alle Ihre Flow-Logs angezeigt.

```
aws ec2 describe-flow-logs
```

Ausgabe:

```
{
  "FlowLogs": [
    {
      "CreationTime": "2018-02-21T13:22:12.644Z",
      "DeliverLogsPermissionArn": "arn:aws:iam::123456789012:role/flow-logs-role",
      "DeliverLogsStatus": "SUCCESS",
      "FlowLogId": "fl-aabbccdd112233445",
      "MaxAggregationInterval": 600,
      "FlowLogStatus": "ACTIVE",
      "LogGroupName": "FlowLogGroup",
      "ResourceId": "subnet-12345678901234567",
      "TrafficType": "ALL",
      "LogDestinationType": "cloud-watch-logs",
      "LogFormat": "${version} ${account-id} ${interface-id} ${srcaddr} ${dstaddr} ${srcport} ${dstport} ${protocol} ${packets} ${bytes} ${start} ${end} ${action} ${log-status}"
    },
  ],
}
```



```

    {
      "CreationTime": "2020-02-04T15:22:29.986Z",
      "DeliverLogsStatus": "SUCCESS",
      "FlowLogId": "fl-01234567890123456",
      "MaxAggregationInterval": 60,
      "FlowLogStatus": "ACTIVE",
      "ResourceId": "vpc-00112233445566778",
      "TrafficType": "ACCEPT",
      "LogDestinationType": "s3",
      "LogDestination": "arn:aws:s3::my-flow-log-bucket/custom",
      "LogFormat": "${version} ${vpc-id} ${subnet-id} ${instance-id}
${interface-id} ${account-id} ${type} ${srcaddr} ${dstaddr} ${srcport}
${dstport} ${pkt-srcaddr} ${pkt-dstaddr} ${protocol} ${bytes} ${packets}
${start} ${end} ${action} ${tcp-flags} ${log-status}"
    }
  ]
}

```

Beispiel 2: Um eine Teilmenge Ihrer Flow-Logs zu beschreiben

Das folgende `describe-flow-logs` Beispiel verwendet einen Filter, um Details nur für die Flow-Logs anzuzeigen, die sich in der angegebenen Protokollgruppe in Amazon CloudWatch Logs befinden.

```

aws ec2 describe-flow-logs \
  --filter "Name=log-group-name,Values=MyFlowLogs"

```

- API-Einheiten finden Sie [DescribeFlowLogs](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt ein oder mehrere Flow-Logs mit dem Protokollzieltyp 's3'

```

Get-EC2FlowLog -Filter @{"Name="log-destination-type";Values="s3"}

```

Ausgabe:

```

CreationTime           : 2/25/2019 9:07:36 PM
DeliverLogsErrorMessage :
DeliverLogsPermissionArn :

```

```

DeliverLogsStatus      : SUCCESS
FlowLogId              : fl-01b2e3d45f67f8901
FlowLogStatus          : ACTIVE
LogDestination         : arn:aws:s3:::my-bucket-dd-tata
LogDestinationType    : s3
LogGroupName           :
ResourceId              : eni-01d2dda3456b7e890
TrafficType            : ALL

```

- API-Einheiten finden Sie unter [DescribeFlowLogs AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeHostReservationOfferings** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung von `DescribeHostReservationOfferings`.

CLI

AWS CLI

Um die Reservierungsangebote für dedizierte Hosts zu beschreiben

In diesem Beispiel werden die Dedicated-Host-Reservierungen für die M4-Instanzfamilie beschrieben, die käuflich erworben werden können.

Befehl:

```
aws ec2 describe-host-reservation-offerings --filter Name=instance-family,Values=m4
```

Ausgabe:

```
{
  "OfferingSet": [
    {
      "HourlyPrice": "1.499",
      "OfferingId": "hro-03f707bf363b6b324",
      "InstanceFamily": "m4",

```

```
    "PaymentOption": "NoUpfront",
    "UpfrontPrice": "0.000",
    "Duration": 31536000
  },
  {
    "HourlyPrice": "1.045",
    "OfferingId": "hro-0ef9181cabdef7a02",
    "InstanceFamily": "m4",
    "PaymentOption": "NoUpfront",
    "UpfrontPrice": "0.000",
    "Duration": 94608000
  },
  {
    "HourlyPrice": "0.714",
    "OfferingId": "hro-04567a15500b92a51",
    "InstanceFamily": "m4",
    "PaymentOption": "PartialUpfront",
    "UpfrontPrice": "6254.000",
    "Duration": 31536000
  },
  {
    "HourlyPrice": "0.484",
    "OfferingId": "hro-0d5d7a9d23ed7fbfe",
    "InstanceFamily": "m4",
    "PaymentOption": "PartialUpfront",
    "UpfrontPrice": "12720.000",
    "Duration": 94608000
  },
  {
    "HourlyPrice": "0.000",
    "OfferingId": "hro-05da4108ca998c2e5",
    "InstanceFamily": "m4",
    "PaymentOption": "AllUpfront",
    "UpfrontPrice": "23913.000",
    "Duration": 94608000
  },
  {
    "HourlyPrice": "0.000",
    "OfferingId": "hro-0a9f9be3b95a3dc8f",
    "InstanceFamily": "m4",
    "PaymentOption": "AllUpfront",
    "UpfrontPrice": "12257.000",
    "Duration": 31536000
  }
}
```

```
]
}
```

- API-Einheiten finden Sie unter [DescribeHostReservationOfferings AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Dedicated Host-Reservierungen beschrieben, die für den angegebenen Filter „Instance-Familie“ erworben werden können, wobei der Filter „PaymentOption“ lautet NoUpfront

```
Get-EC2HostReservationOffering -Filter @{Name="instance-family";Values="m4"} |
Where-Object PaymentOption -eq NoUpfront
```

Ausgabe:

```
CurrencyCode      :
Duration          : 94608000
HourlyPrice       : 1.307
InstanceFamily    : m4
OfferingId        : hro-0c1f234567890d9ab
PaymentOption     : NoUpfront
UpfrontPrice      : 0.000

CurrencyCode      :
Duration          : 31536000
HourlyPrice       : 1.830
InstanceFamily    : m4
OfferingId        : hro-04ad12aaaf34b5a67
PaymentOption     : NoUpfront
UpfrontPrice      : 0.000
```

- API-Einheiten finden Sie unter [DescribeHostReservationOfferings AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeHosts** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeHosts`.

CLI

AWS CLI

Um Details zu Dedicated Hosts anzuzeigen

Im folgenden `describe-hosts` Beispiel werden Details zu den available Dedicated Hosts in Ihrem AWS Konto angezeigt.

```
aws ec2 describe-hosts --filter "Name=state,Values=available"
```

Ausgabe:

```
{
  "Hosts": [
    {
      "HostId": "h-07879acf49EXAMPLE",
      "Tags": [
        {
          "Value": "production",
          "Key": "purpose"
        }
      ],
      "HostProperties": {
        "Cores": 48,
        "TotalVCpus": 96,
        "InstanceType": "m5.large",
        "Sockets": 2
      },
      "Instances": [],
      "State": "available",
      "AvailabilityZone": "eu-west-1a",
      "AvailableCapacity": {
        "AvailableInstanceCapacity": [
```

```
        {
            "AvailableCapacity": 48,
            "InstanceType": "m5.large",
            "TotalCapacity": 48
        }
    ],
    "AvailableVCpus": 96
},
"HostRecovery": "on",
"AllocationTime": "2019-08-19T08:57:44.000Z",
"AutoPlacement": "off"
}
]
}
```

Weitere Informationen finden Sie unter [Viewing Dedicated Hosts](#) im Amazon Elastic Compute Cloud-Benutzerhandbuch für Linux-Instances.

- API-Einheiten finden Sie [DescribeHosts](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt die EC2 Host-Details zurück

```
Get-EC2Host
```

Ausgabe:

```
AllocationTime      : 3/23/2019 4:55:22 PM
AutoPlacement       : off
AvailabilityZone    : eu-west-1b
AvailableCapacity   : Amazon.EC2.Model.AvailableCapacity
ClientToken         :
HostId              : h-01e23f4cd567890f1
HostProperties       : Amazon.EC2.Model.HostProperties
HostReservationId   :
Instances           : {}
ReleaseTime         : 1/1/0001 12:00:00 AM
State               : available
Tags                : {}
```

Beispiel 2: In diesem Beispiel wird nach dem Host AvailableInstanceCapacity h-01e23f4cd567899f1 abgefragt

```
Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty AvailableCapacity | Select-Object -expand AvailableInstanceCapacity
```

Ausgabe:

```
AvailableCapacity InstanceType TotalCapacity
-----
11                m4.xlarge    11
```

- API-Einheiten [DescribeHosts](#) finden AWS Tools for PowerShell Sie unter Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeIamInstanceProfileAssociations** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DescribeIamInstanceProfileAssociations`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erstellen und Verwalten eines ausfallsicheren Services](#)

.NET

AWS SDK for .NET

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Get the instance profile association data for an instance.
/// </summary>
/// <param name="instanceId">The Id of the instance.</param>
/// <returns>Instance profile associations data.</returns>
public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
{
    try
    {
        var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
            new DescribeIamInstanceProfileAssociationsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new("instance-id", new List<string>() { instanceId })
                },
            });
        return response.IamInstanceProfileAssociations[0];
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
        {
            _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
        throw;
    }
}
```

- API-Einheiten finden Sie [DescribeIamInstanceProfileAssociations](#) unter AWS SDK for .NET API-Referenz.

CLI

AWS CLI

Um die Zuordnungen von IAM Instanzprofilen zu beschreiben

In diesem Beispiel werden alle Ihre IAM Instanzprofilzuordnungen beschrieben.

Befehl:

```
aws ec2 describe-iam-instance-profile-associations
```

Ausgabe:

```
{
  "IamInstanceProfileAssociations": [
    {
      "InstanceId": "i-09eb09efa73ec1dee",
      "State": "associated",
      "AssociationId": "iip-assoc-0db249b1f25fa24b8",
      "IamInstanceProfile": {
        "Id": "AIPAJVQN4F5WVLGCJDRGM",
        "Arn": "arn:aws:iam::123456789012:instance-profile/admin-role"
      }
    },
    {
      "InstanceId": "i-0402909a2f4dffd14",
      "State": "associating",
      "AssociationId": "iip-assoc-0d1ec06278d29f44a",
      "IamInstanceProfile": {
        "Id": "AGJAJVQN4F5WVLGCJABCM",
        "Arn": "arn:aws:iam::123456789012:instance-profile/user1-role"
      }
    }
  ]
}
```

- API-Einheiten finden Sie [DescribeIamInstanceProfileAssociations](#) in der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
const ec2Client = new EC2Client({});
const { IamInstanceProfileAssociations } = await ec2Client.send(
  new DescribeIamInstanceProfileAssociationsCommand({
    Filters: [
      { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
    ],
  }),
);
```

- API-Einheiten finden Sie [DescribeIamInstanceProfileAssociations](#) unter AWS SDK for JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
```

```
resource_prefix: str,
inst_type: str,
ami_param: str,
autoscaling_client: boto3.client,
ec2_client: boto3.client,
ssm_client: boto3.client,
iam_client: boto3.client,
):
    """
    Initializes the AutoScaler class with the necessary parameters.

    :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
    :param inst_type: The type of EC2 instance to create, such as t3.micro.
    :param ami_param: The Systems Manager parameter used to look up the AMI
that is created.
    :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
    :param ec2_client: A Boto3 EC2 client.
    :param ssm_client: A Boto3 Systems Manager client.
    :param iam_client: A Boto3 IAM client.
    """
    self.inst_type = inst_type
    self.ami_param = ami_param
    self.autoscaling_client = autoscaling_client
    self.ec2_client = ec2_client
    self.ssm_client = ssm_client
    self.iam_client = iam_client
    sts_client = boto3.client("sts")
    self.account_id = sts_client.get_caller_identity()["Account"]

    self.key_pair_name = f"{resource_prefix}-key-pair"
    self.launch_template_name = f"{resource_prefix}-template-"
    self.group_name = f"{resource_prefix}-group"

    # Happy path
    self.instance_policy_name = f"{resource_prefix}-pol"
    self.instance_role_name = f"{resource_prefix}-role"
    self.instance_profile_name = f"{resource_prefix}-prof"

    # Failure mode
    self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
    self.bad_creds_role_name = f"{resource_prefix}-bc-role"
    self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"
```

```

def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
    """
    Gets data about the profile associated with an instance.

    :param instance_id: The ID of the instance to look up.
    :return: The profile data.
    """
    try:
        response =
self.ec2_client.describe_iam_instance_profile_associations(
            Filters=[{"Name": "instance-id", "Values": [instance_id]}]
        )
        if not response["IamInstanceProfileAssociations"]:
            log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
    except ClientError as err:
        log.error(
            f"Failed to retrieve instance profile for instance
{instance_id}."
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidInstanceID.NotFound":
            log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")

```

- API-Einheiten finden Sie unter [DescribeIamInstanceProfileAssociations](#) Python (Boto3) API-Referenz. AWS SDK

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeIdFormat** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung von `DescribeIdFormat`.

CLI

AWS CLI

Beispiel 1: Um das ID-Format einer Ressource zu beschreiben

Das folgende `describe-id-format` Beispiel beschreibt das ID-Format für Sicherheitsgruppen.

```
aws ec2 describe-id-format \  
  --resource security-group
```

In der folgenden Beispielausgabe gibt der `Deadline` Wert an, dass die Frist für den dauerhaften Wechsel dieses Ressourcentyps vom kurzen ID-Format zum langen ID-Format am 15. August 2018 um 00:00 Uhr UTC abgelaufen ist.

```
{  
  "Statuses": [  
    {  
      "Deadline": "2018-08-15T00:00:00.000Z",  
      "Resource": "security-group",  
      "UseLongIds": true  
    }  
  ]  
}
```

Beispiel 2: Um das ID-Format für alle Ressourcen zu beschreiben

Das folgende `describe-id-format` Beispiel beschreibt das ID-Format für alle Ressourcentypen. Alle Ressourcentypen, die das kurze ID-Format unterstützten, wurden auf das lange ID-Format umgestellt.

```
aws ec2 describe-id-format
```

- API-Einheiten finden Sie [Beschreibung des ID-Formats](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das ID-Format für den angegebenen Ressourcentyp.

```
Get-EC2IdFormat -Resource instance
```

Ausgabe:

```
Resource      UseLongIds
-----
instance      False
```

Beispiel 2: Dieses Beispiel beschreibt die ID-Formate für alle Ressourcentypen, die Longer unterstützen IDs.

```
Get-EC2IdFormat
```

Ausgabe:

```
Resource      UseLongIds
-----
reservation   False
instance       False
```

- API-Einheiten finden Sie unter [DescribeIdFormat AWS Tools for PowerShell Cmdlet](#)-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeIdentityIdFormat** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeIdentityIdFormat`.

CLI

AWS CLI

Um das ID-Format für eine IAM Rolle zu beschreiben

Das folgende `describe-identity-id-format` Beispiel beschreibt das ID-Format, das von Instances empfangen wird, die von der IAM Rolle `EC2Role` in Ihrem AWS Konto erstellt wurden.

```
aws ec2 describe-identity-id-format \  
  --principal-arn arn:aws:iam::123456789012:role/my-iam-role \  
  --resource instance
```

Die folgende Ausgabe zeigt, dass Instanzen, die von dieser Rolle erstellt wurden, IDs im Long-ID-Format empfangen werden.

```
{  
  "Statuses": [  
    {  
      "Deadline": "2016-12-15T00:00:00Z",  
      "Resource": "instance",  
      "UseLongIds": true  
    }  
  ]  
}
```

Um das ID-Format für einen IAM Benutzer zu beschreiben

Das folgende `describe-identity-id-format` Beispiel beschreibt das ID-Format, das von Snapshots empfangen wird, die vom IAM Benutzer `AdminUser` in Ihrem AWS Konto erstellt wurden.

```
aws ec2 describe-identity-id-format \  
  --principal-arn arn:aws:iam::123456789012:user/AdminUser \  
  --resource snapshot
```

Aus der Ausgabe geht hervor, dass von diesem Benutzer erstellte Schnappschüsse IDs im Long-ID-Format empfangen werden.

```
{  
  "Statuses": [  
    {  
      "Deadline": "2016-12-15T00:00:00Z",  
      "Resource": "snapshot",  
      "UseLongIds": true  
    }  
  ]  
}
```

- API-Einheiten finden Sie [DescribeIdentityIdFormat](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel gibt das ID-Format für die Ressource 'image' für die angegebene Rolle zurück

```
Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -
Resource image
```

Ausgabe:

```
Deadline                Resource UseLongIds
-----                -
8/2/2018 11:30:00 PM image    True
```

- API-Einheiten finden Sie unter [DescribeIdentityIdFormat AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeImageAttribute** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeImageAttribute`.

CLI

AWS CLI

Um die Startberechtigungen für einen zu beschreiben AMI

In diesem Beispiel werden die Startberechtigungen für die angegebene Datei beschrieben AMI.

Befehl:

```
aws ec2 describe-image-attribute --image-id ami-5731123e --
attribute LaunchPermission
```

Ausgabe:


```
{
  "LaunchPermissions": [
    {
      "UserId": "123456789012"
    }
  ],
  "ImageId": "ami-5731123e",
}
```

Um die Produktcodes für eine zu beschreiben AMI

In diesem Beispiel werden die Produktcodes für die angegebenen Produkte beschriebenAMI. Beachten Sie, dass AMI es hier keine Produktcodes gibt.

Befehl:

```
aws ec2 describe-image-attribute --image-id ami-5731123e --attribute productCodes
```

Ausgabe:

```
{
  "ProductCodes": [],
  "ImageId": "ami-5731123e",
}
```

- APIEinzelheiten finden Sie [DescribeImageAttribute](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Beschreibung für das angegebene Objekt abgerufenAMI.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

Ausgabe:

```
BlockDeviceMappings : {}
Description           : My image description
```

```
ImageId      : ami-12345678
KernelId     :
LaunchPermissions : {}
ProductCodes : {}
RamdiskId    :
SriovNetSupport :
```

Beispiel 2: In diesem Beispiel werden die Startberechtigungen für die angegebene Datei abgerufenAMI.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

Ausgabe:

```
BlockDeviceMappings : {}
Description          :
ImageId              : ami-12345678
KernelId             :
LaunchPermissions    : {all}
ProductCodes         : {}
RamdiskId            :
SriovNetSupport      :
```

Beispiel 3: In diesem Beispiel wird getestet, ob Enhanced Networking aktiviert ist.

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

Ausgabe:

```
BlockDeviceMappings : {}
Description          :
ImageId              : ami-12345678
KernelId             :
LaunchPermissions    : {}
ProductCodes         : {}
RamdiskId            :
SriovNetSupport      : simple
```

- APIEinzelheiten finden Sie unter [DescribeImageAttribute AWS Tools for PowerShellCmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeImages** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DescribeImages`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#####
# function ec2_describe_images
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
# images.
#
# Parameters:
#     -i image_ids - A space-separated list of image IDs (optional).
#     -h - Display help.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_describe_images() {
    local image_ids response
    local option OPTARG # Required to use getopt command in a function.
```

```
# bashsupport disable=BP5008
function usage() {
    echo "function ec2_describe_images"
    echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
images."
    echo "  -i image_ids - A space-separated list of image IDs (optional)."
    echo "  -h - Display help."
    echo ""
}

# Retrieve the calling parameters.
while getopts "i:h" option; do
    case "${option}" in
        i) image_ids="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

local aws_cli_args=()

if [[ -n "$image_ids" ]]; then
    # shellcheck disable=SC2206
    aws_cli_args+=("--image-ids" $image_ids)
fi

response=$(aws ec2 describe-images \
    "${aws_cli_args[@]}" \
    --query 'Images[*].[Description,Architecture,ImageId]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-images operation failed.$response"
    return 1
}

echo "$response"
```

```

return 0
}

```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then
        errecho " The service returned an error."
    }
}

```

```
elif [ "$err_code" == 255 ]; then
    errecho " 255 is a catch-all error."
fi

return 0
}
```

- API-Einheiten finden Sie [Beschreiben von Images](#) in der AWS CLI Befehlsreferenz.

CLI

AWS CLI

Beispiel 1: Um ein zu beschreiben AMI

Das folgende `describe-images` Beispiel beschreibt die AMI in der angegebenen Region angegebene.

```
aws ec2 describe-images \
  --region us-east-1 \
  --image-ids ami-1234567890EXAMPLE
```

Ausgabe:

```
{
  "Images": [
    {
      "VirtualizationType": "hvm",
      "Description": "Provided by Red Hat, Inc.",
      "PlatformDetails": "Red Hat Enterprise Linux",
      "EnaSupport": true,
      "Hypervisor": "xen",
      "State": "available",
      "SriovNetSupport": "simple",
      "ImageId": "ami-1234567890EXAMPLE",
      "UsageOperation": "RunInstances:0010",
      "BlockDeviceMappings": [
        {
          "DeviceName": "/dev/sda1",
          "Ebs": {
            "SnapshotId": "snap-111222333444aaabb",
```

```

        "DeleteOnTermination": true,
        "VolumeType": "gp2",
        "VolumeSize": 10,
        "Encrypted": false
    }
}
],
"Architecture": "x86_64",
"ImageLocation": "123456789012/RHEL-8.0.0_HVM-20190618-x86_64-1-
Hourly2-GP2",
"RootDeviceType": "ebs",
"OwnerId": "123456789012",
"RootDeviceName": "/dev/sda1",
"CreationDate": "2019-05-10T13:17:12.000Z",
"Public": true,
"ImageType": "machine",
"Name": "RHEL-8.0.0_HVM-20190618-x86_64-1-Hourly2-GP2"
}
]
}

```

Weitere Informationen finden Sie unter [Amazon Machine Images \(AMI\)](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 2: Zur Beschreibung AMIs anhand von Filtern

Das folgende `describe-images` Beispiel beschreibt Windows, das von Amazon AMIs bereitgestellt und von Amazon unterstützt wird EBS.

```

aws ec2 describe-images \
  --owners amazon \
  --filters "Name=platform,Values=windows" "Name=root-device-type,Values=ebs"

```

Ein Beispiel für die Ausgabe von `describe-images` finden Sie in Beispiel 1.

Weitere Beispiele für die Verwendung von Filtern finden Sie unter [Ressourcen auflisten und filtern](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 3: Zur Beschreibung AMIs anhand von Stichwörtern

Das folgende `describe-images` Beispiel beschreibt alle AMIs, die das Tag `habenType=Custom`. Das Beispiel verwendet den `--query` Parameter, um nur die anzuzeigen AMIIDs.

```
aws ec2 describe-images \
  --filters "Name=tag:Type,Values=Custom" \
  --query 'Images[*].[ImageId]' \
  --output text
```

Ausgabe:

```
ami-1234567890EXAMPLE
ami-0abcdef1234567890
```

Weitere Beispiele für die Verwendung von Tag-Filtern finden Sie unter [Arbeiten mit Tags](#) im EC2Amazon-Benutzerhandbuch.

- API-Einheiten finden Sie [DescribeImages](#) in der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import { EC2Client, paginateDescribeImages } from "@aws-sdk/client-ec2";

/**
 * Describes the specified images (AMIs, AKIs, and ARIs) available to you or all
 * of the images available to you.
 * @param {{ architecture: string, pageSize: number }} options
 */
export const main = async ({ architecture, pageSize }) => {
  pageSize = Number.parseInt(pageSize);
  const client = new EC2Client({});

  // The paginate function is a wrapper around the base command.
  const paginator = paginateDescribeImages(
    // Without limiting the page size, this call can take a long time. pageSize
    // is just sugar for
    // the MaxResults property in the base command.
```



```
{ client, pageSize },
{
  // There are almost 70,000 images available. Be specific with your
  filtering
  // to increase efficiency.
  // See https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/clients/
  client-ec2/interfaces/describeimagescommandinput.html#filters
  Filters: [{ Name: "architecture", Values: [architecture] }],
},
);

/**
 * @type {import('@aws-sdk/client-ec2').Image[]}
 */
const images = [];
let recordsScanned = 0;

try {
  for await (const page of paginator) {
    recordsScanned += pageSize;
    if (page.Images.length) {
      images.push(...page.Images);
      break;
    }
    console.log(
      `No matching image found yet. Searched ${recordsScanned} records.`,
    );
  }

  if (images.length) {
    console.log(
      `Found ${images.length} images:\n\n${images.map((image) =>
image.Name).join("\n")}\n`,
    );
  } else {
    console.log(
      `No matching images found. Searched ${recordsScanned} records.\n`,
    );
  }

  return images;
} catch (caught) {
  if (caught instanceof Error && caught.name === "InvalidParameterValue") {
    console.warn(`${caught.message}`);
  }
}
```

```
        return [];  
    }  
    throw caught;  
}  
};
```

- API-Einheiten finden Sie [Beschreiben von Images](#) unter AWS SDK for JavaScript API-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebenen AMI.

```
Get-EC2Image -ImageId ami-12345678
```

Ausgabe:

```
Architecture      : x86_64  
BlockDeviceMappings : {/dev/xvda}  
CreationDate      : 2014-10-20T00:56:28.000Z  
Description       : My image  
Hypervisor        : xen  
ImageId           : ami-12345678  
ImageLocation     : 123456789012/my-image  
ImageOwnerAlias   :  
ImageType         : machine  
KernelId         :  
Name              : my-image  
OwnerId           : 123456789012  
Platform         :  
ProductCodes     : {}  
Public            : False  
RamdiskId        :  
RootDeviceName   : /dev/xvda  
RootDeviceType   : ebs  
SriovNetSupport  : simple  
State             : available  
StateReason      :  
Tags              : {Name}  
VirtualizationType : hvm
```

Beispiel 2: Dieses Beispiel beschreibt AMIs das, was Sie besitzen.

```
Get-EC2Image -owner self
```

Beispiel 3: Dieses Beispiel beschreibt die ÖffentlichkeitAMIs, die Microsoft Windows Server ausführt.

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

Beispiel 4: Dieses Beispiel beschreibt alle Besucher AMIs in der Region „us-west-2“.

```
Get-EC2Image -Region us-west-2
```

- APIEinzelheiten finden Sie unter [DescribeImages AWS Tools for PowerShell](#) Cmdlet-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
```

```

        access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
objects. These are high-level objects that
        wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def get_images(self, image_ids: List[str]) -> List[Dict[str, Any]]:
        """
        Gets information about Amazon Machine Images (AMIs) from a list of AMI
IDs.


        :param image_ids: The list of AMI IDs to look up.
        :return: A list of dictionaries representing the requested AMIs.
        """
        try:
            response = self.ec2_client.describe_images(ImageIds=image_ids)
            images = response["Images"]
        except ClientError as err:
            logger.error(f"Failed to stop AMI(s): {','.join(map(str,
image_ids))}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAMIID.NotFound":
                logger.error("One or more of the AMI IDs does not exist.")
            raise
        return images

```

- APIEinzelheiten finden Sie unter [DescribeImages](#)Python (Boto3) API -Referenz.AWS SDK

Rust

SDK für Rust

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
pub async fn list_images(&self, ids: Vec<Parameter>) -> Result<Vec<Image>,
EC2Error> {
    let image_ids = ids.into_iter().filter_map(|p| p.value).collect();
    let output = self
        .client
        .describe_images()
        .set_image_ids(Some(image_ids))
        .send()
        .await?;

    let images = output.images.unwrap_or_default();
    if images.is_empty() {
        Err(EC2Error::new("No images for selected AMIs"))
    } else {
        Ok(images)
    }
}
```

Verwenden Sie die Funktion `list_images` mit SSM, um sie je nach Umgebung einzuschränken. Weitere Informationen dazu finden Sie unter SSM https://docs.aws.amazon.com/systems-manager/latest/userguide/example_GetParameters_ssm_section.html.

```
async fn find_image(&mut self) -> Result<ScenarioImage, EC2Error> {
    let params: Vec<Parameter> = self
        .ssm
        .list_path("/aws/service/ami-amazon-linux-latest")
        .await
        .map_err(|e| e.add_message("Could not find parameters for available
images"))?
        .into_iter()
```

```

        .filter(|param| param.name().is_some_and(|name|
name.contains("amzn2")))
        .collect();
    let amzn2_images: Vec<ScenarioImage> = self
        .ec2
        .list_images(params)
        .await
        .map_err(|e| e.add_message("Could not find images"))?
        .into_iter()
        .map(ScenarioImage::from)
        .collect();
    println!("We will now create an instance from an Amazon Linux 2 AMI");
    let ami = self.util.select_scenario_image(amzn2_images)?;
    Ok(ami)
}

```

- API-Einheiten finden Sie [DescribeImages](#) in AWS SDK der Rust-Referenz. API

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeImportImageTasks** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeImportImageTasks`.

CLI

AWS CLI

Um eine Aufgabe zum Importieren eines Bilds zu überwachen

Im folgenden `describe-import-image-tasks` Beispiel wird der Status der angegebenen Aufgabe zum Importieren von Bildern überprüft.

```
aws ec2 describe-import-image-tasks \
  --import-task-ids import-ami-1234567890abcdef0
```

Ausgabe für eine Aufgabe zum Importieren von Bildern, die gerade ausgeführt wird.

```
{
```

```

    "ImportImageTasks": [
      {
        "ImportTaskId": "import-ami-1234567890abcdef0",
        "Progress": "28",
        "SnapshotDetails": [
          {
            "DiskImageSize": 705638400.0,
            "Format": "ova",
            "Status": "completed",
            "UserBucket": {
              "S3Bucket": "my-import-bucket",
              "S3Key": "vms/my-server-vm.ova"
            }
          }
        ],
        "Status": "active",
        "StatusMessage": "converting"
      }
    ]
  }
}

```

Ausgabe für eine Aufgabe zum Importieren von Bildern, die abgeschlossen ist. Die ID des Ergebnisses AMI wird von `bereitgestelltImageId`.

```

{
  "ImportImageTasks": [
    {
      "ImportTaskId": "import-ami-1234567890abcdef0",
      "ImageId": "ami-1234567890abcdef0",
      "SnapshotDetails": [
        {
          "DiskImageSize": 705638400.0,
          "Format": "ova",
          "SnapshotId": "snap-1234567890abcdef0"
          "Status": "completed",
          "UserBucket": {
            "S3Bucket": "my-import-bucket",
            "S3Key": "vms/my-server-vm.ova"
          }
        }
      ],
      "Status": "completed"
    }
  ]
}

```

```
]
}
```

- API-Einheiten finden Sie [DescribeImportImageTasks](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Bildimportaufgabe.

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

Ausgabe:

```
Architecture      : x86_64
Description       : Windows Image 2
Hypervisor        :
ImageId           : ami-1a2b3c4d
ImportTaskId      : import-ami-hgfedcba
LicenseType       : AWS
Platform          : Windows
Progress          :
SnapshotDetails  : {/dev/sda1}
Status            : completed
StatusMessage     :
```

Beispiel 2: In diesem Beispiel werden alle Ihre Bildimportaufgaben beschrieben.

```
Get-EC2ImportImageTask
```

Ausgabe:

```
Architecture      :
Description       : Windows Image 1
Hypervisor        :
ImageId           :
ImportTaskId      : import-ami-abcdefgh
LicenseType       : AWS
Platform          : Windows
```



```

Progress      :
SnapshotDetails : {}
Status       : deleted
StatusMessage : User initiated task cancelation

Architecture : x86_64
Description  : Windows Image 2
Hypervisor   :
ImageId      : ami-1a2b3c4d
ImportTaskId : import-ami-hgfedcba
LicenseType  : AWS
Platform     : Windows
Progress     :
SnapshotDetails : {/dev/sda1}
Status      : completed
StatusMessage :

```

- API-Einheiten finden Sie unter [DescribeImportImageTasks AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeImportSnapshotTasks** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung von `DescribeImportSnapshotTasks`.

CLI

AWS CLI

Um eine Aufgabe zum Importieren eines Snapshots zu überwachen

Im folgenden `describe-import-snapshot-tasks` Beispiel wird der Status der angegebenen Import-Snapshot-Aufgabe überprüft.

```

aws ec2 describe-import-snapshot-tasks \
  --import-task-ids import-snap-1234567890abcdef0

```

Ausgabe für eine Snapshot-Importaufgabe, die gerade ausgeführt wird:

```
{
  "ImportSnapshotTasks": [
    {
      "Description": "My server VMDK",
      "ImportTaskId": "import-snap-1234567890abcdef0",
      "SnapshotTaskDetail": {
        "Description": "My server VMDK",
        "DiskImageSize": "705638400.0",
        "Format": "VMDK",
        "Progress": "42",
        "Status": "active",
        "StatusMessage": "downloading/convertng",
        "UserBucket": {
          "S3Bucket": "my-import-bucket",
          "S3Key": "vms/my-server-vm.vmdk"
        }
      }
    }
  ]
}
```

Ausgabe für eine abgeschlossene Import-Snapshot-Aufgabe. Die ID des resultierenden Snapshots wird von `preparedSnapshotId`.

```
{
  "ImportSnapshotTasks": [
    {
      "Description": "My server VMDK",
      "ImportTaskId": "import-snap-1234567890abcdef0",
      "SnapshotTaskDetail": {
        "Description": "My server VMDK",
        "DiskImageSize": "705638400.0",
        "Format": "VMDK",
        "SnapshotId": "snap-1234567890abcdef0"
        "Status": "completed",
        "UserBucket": {
          "S3Bucket": "my-import-bucket",
          "S3Key": "vms/my-server-vm.vmdk"
        }
      }
    }
  ]
}
```

```
}

```

- API-Einheiten finden Sie [Beschreiben von Import-Snapshot-Aufgaben](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Snapshot-Importaufgabe.

```
Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh
```

Ausgabe:

Description	ImportTaskId	SnapshotTaskDetail
-----	-----	-----
Disk Image Import 1	import-snap-abcdefgh	Amazon.EC2.Model.SnapshotTaskDetail

Beispiel 2: In diesem Beispiel werden alle Ihre Snapshot-Importaufgaben beschrieben.

```
Get-EC2ImportSnapshotTask
```

Ausgabe:

Description	ImportTaskId	SnapshotTaskDetail
-----	-----	-----
Disk Image Import 1	import-snap-abcdefgh	Amazon.EC2.Model.SnapshotTaskDetail
Disk Image Import 2	import-snap-hgfedcba	Amazon.EC2.Model.SnapshotTaskDetail

- API-Einheiten finden Sie unter [Beschreiben von Import-Snapshot-Aufgaben AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeInstanceAttribute** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeInstanceAttribute`.

CLI

AWS CLI

Um den Instanztyp zu beschreiben

Dieses Beispiel beschreibt den Instanztyp der angegebenen Instanz.

Befehl:

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --  
attribute instanceType
```

Ausgabe:

```
{  
  "InstanceId": "i-1234567890abcdef0"  
  "InstanceType": {  
    "Value": "t1.micro"  
  }  
}
```

Um das `disableApiTermination` Attribut zu beschreiben

Dieses Beispiel beschreibt das `disableApiTermination` Attribut der angegebenen Instanz.

Befehl:

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --  
attribute disableApiTermination
```

Ausgabe:

```
{
```

```
"InstanceId": "i-1234567890abcdef0"
  "DisableApiTermination": {
    "Value": "false"
  }
}
```

Um die Blockgerätezuzuweisung für eine Instanz zu beschreiben

Dieses Beispiel beschreibt das `blockDeviceMapping` Attribut der angegebenen Instanz.

Befehl:

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --
attribute blockDeviceMapping
```

Ausgabe:

```
{
  "InstanceId": "i-1234567890abcdef0"
  "BlockDeviceMappings": [
    {
      "DeviceName": "/dev/sda1",
      "Ebs": {
        "Status": "attached",
        "DeleteOnTermination": true,
        "VolumeId": "vol-049df61146c4d7901",
        "AttachTime": "2013-05-17T22:42:34.000Z"
      }
    },
    {
      "DeviceName": "/dev/sdf",
      "Ebs": {
        "Status": "attached",
        "DeleteOnTermination": false,
        "VolumeId": "vol-049df61146c4d7901",
        "AttachTime": "2013-09-10T23:07:00.000Z"
      }
    }
  ],
}
```

- API-Einheiten finden Sie [DescribeInstanceAttribute](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt den Instanztyp der angegebenen Instanz.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

Ausgabe:

```
InstanceType           : t2.micro
```

Beispiel 2: In diesem Beispiel wird beschrieben, ob Enhanced Networking für die angegebene Instanz aktiviert ist.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Ausgabe:

```
SriovNetSupport       : simple
```

Beispiel 3: In diesem Beispiel werden die Sicherheitsgruppen für die angegebene Instance beschrieben.

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

Ausgabe:

```
GroupId
-----
sg-12345678
sg-45678901
```

Beispiel 4: In diesem Beispiel wird beschrieben, ob die EBS Optimierung für die angegebene Instanz aktiviert ist.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Ausgabe:

```
EbsOptimized : False
```

Beispiel 5: Dieses Beispiel beschreibt das Attribut `disableApiTermination` der angegebenen Instanz.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

Ausgabe:

```
DisableApiTermination : False
```

Beispiel 6: Dieses Beispiel beschreibt das Attribut `instanceInitiatedShutdownBehavior` der angegebenen Instanz.

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute  
instanceInitiatedShutdownBehavior
```

Ausgabe:

```
InstanceInitiatedShutdownBehavior : stop
```

- API-Einheiten finden Sie unter [DescribeInstanceAttribute AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeInstanceStatus** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DescribeInstanceStatus`.

CLI

AWS CLI

So beschreiben Sie den Status einer Instanz

Das folgende `describe-instance-status`-Beispiel beschreibt den aktuellen Status der angegebenen Instance.

```
aws ec2 describe-instance-status \  
  --instance-ids i-1234567890abcdef0
```

Ausgabe:

```
{  
  "InstanceStatuses": [  
    {  
      "InstanceId": "i-1234567890abcdef0",  
      "InstanceState": {  
        "Code": 16,  
        "Name": "running"  
      },  
      "AvailabilityZone": "us-east-1d",  
      "SystemStatus": {  
        "Status": "ok",  
        "Details": [  
          {  
            "Status": "passed",  
            "Name": "reachability"  
          }  
        ]  
      },  
      "InstanceStatus": {  
        "Status": "ok",  
        "Details": [  
          {  
            "Status": "passed",  
            "Name": "reachability"  
          }  
        ]  
      }  
    }  
  ]  
}
```

Weitere Informationen finden Sie unter [Überwachen des Status Ihrer Instances](#) im EC2Amazon-Benutzerhandbuch.

- API-Einheiten finden Sie [DescribeInstanceStatus](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt den Status der angegebenen Instanz.

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

Ausgabe:

```
AvailabilityZone : us-west-2a
Events           : {}
InstanceId       : i-12345678
InstanceState    : Amazon.EC2.Model.InstanceState
Status          : Amazon.EC2.Model.InstanceStatusSummary
SystemStatus     : Amazon.EC2.Model.InstanceStatusSummary
```

```
$status = Get-EC2InstanceStatus -InstanceId i-12345678
$status.InstanceState
```

Ausgabe:

```
Code    Name
----    -
16      running
```

```
$status.Status
```

Ausgabe:

```
Details      Status
-----      -
{reachability} ok
```

```
$status.SystemStatus
```

Ausgabe:

```
Details      Status
-----      -
```

```
{reachability}    ok
```

- API-Einheiten finden Sie unter [DescribeInstanceStatus AWS Tools for PowerShell Cmdlet-Referenz](#).

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
async fn show_all_events(client: &Client) -> Result<(), Error> {
    let resp = client.describe_regions().send().await.unwrap();

    for region in resp.regions.unwrap_or_default() {
        let reg: &'static str =
            Box::leak(Box::from(region.region_name().unwrap()));
        let region_provider =
            RegionProviderChain::default_provider().or_else(reg);
        let config = aws_config::from_env().region(region_provider).load().await;
        let new_client = Client::new(&config);

        let resp = new_client.describe_instance_status().send().await;

        println!("Instances in region {}: ", reg);
        println!();

        for status in resp.unwrap().instance_statuses() {
            println!(
                "  Events scheduled for instance ID: {}",
                status.instance_id().unwrap_or_default()
            );
            for event in status.events() {
                println!("    Event ID:      {}",
                    event.instance_event_id().unwrap());
                println!("    Description:  {}", event.description().unwrap());
                println!("    Event code:   {}", event.code().unwrap().as_ref());
            }
        }
    }
}
```

```

        println!();
    }
}
}
    ok(())
}

```

- API-Einheiten finden Sie [DescribeInstanceStatus](#) in der AWS SDK API Rust-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeInstanceTypes** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DescribeInstanceTypes`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

/// <summary>
/// Describe the instance types available.
/// </summary>
/// <returns>A list of instance type information.</returns>
public async Task<List<InstanceTypeInfo>>
DescribeInstanceTypes(ArchitectureValues architecture)

```

```
{
    try
    {
        var request = new DescribeInstanceTypesRequest();

        var filters = new List<Filter>
        {
            new Filter("processor-info.supported-architecture",
                new List<string> { architecture.ToString() })
        };
        filters.Add(new Filter("instance-type", new() { "*.micro",
            "*.small" }));

        request.Filters = filters;
        var instanceTypes = new List<InstanceTypeInfo>();

        var paginator = _amazonEC2.Paginators.DescribeInstanceTypes(request);
        await foreach (var instanceType in paginator.InstanceTypes)
        {
            instanceTypes.Add(instanceType);
        }

        return instanceTypes;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidParameterValue")
        {
            _logger.LogError(
                $"Parameters are invalid. Ensure architecture and size
strings conform to DescribeInstanceTypes API reference.");
        }

        throw;
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Couldn't delete the security group because:
{ex.Message}");
        throw;
    }
}
```

- API-Einheiten finden Sie [DescribeInstanceTypes](#) unter AWS SDK for .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#####
# ec2_describe_instance_types
#
# This function describes EC2 instance types filtered by processor architecture
# and optionally by instance type. It takes the following arguments:
#
# -a, --architecture ARCHITECTURE  Specify the processor architecture (e.g.,
# x86_64)
# -t, --type INSTANCE_TYPE          Comma-separated list of instance types (e.g.,
# t2.micro)
# -h, --help                        Show the usage help
#
# The function prints the instance type and supported architecture for each
# matching instance type.
#####
function ec2_describe_instance_types() {
    local architecture=""
    local instance_types=""

    # bashsupport disable=BP5008
    function usage() {
        echo "Usage: ec2_describe_instance_types [-a|--architecture ARCHITECTURE] [-t|--type INSTANCE_TYPE] [-h|--help]"
        echo "  -a, --architecture ARCHITECTURE  Specify the processor architecture (e.g., x86_64)"
        echo "  -t, --type INSTANCE_TYPE          Comma-separated list of instance types (e.g., t2.micro)"
        echo "  -h, --help                        Show this help message"
    }
}
```

```
while [[ $# -gt 0 ]]; do
  case "$1" in
    -a | --architecture)
      architecture="$2"
      shift 2
      ;;
    -t | --type)
      instance_types="$2"
      shift 2
      ;;
    -h | --help)
      usage
      return 0
      ;;
    *)
      echo "Unknown argument: $1"
      return 1
      ;;
  esac
done

if [[ -z "$architecture" ]]; then
  errecho "Error: Architecture not specified."
  usage
  return 1
fi

if [[ -z "$instance_types" ]]; then
  errecho "Error: Instance type not specified."
  usage
  return 1
fi

local tmp_json_file="temp_ec2.json"
echo -n '[
  {
    "Name": "processor-info.supported-architecture",
    "Values": [' >"$tmp_json_file"

local items
IFS=',' read -ra items <<<"$architecture"
local array_size
array_size=${#items[@]}
for i in $(seq 0 $((array_size - 1))); do
```

```

    echo -n '""${items[$i]}""' >>"$tmp_json_file"
    if [[ $i -lt $((array_size - 1)) ]]; then
        echo -n ',' >>"$tmp_json_file"
    fi
done
echo -n ']],
{
    "Name": "instance-type",
    "Values": [' >>"$tmp_json_file"
IFS=', ' read -ra items <<<"$instance_types"
local array_size
array_size=${#items[@]}
for i in $(seq 0 $((array_size - 1))); do
    echo -n '""${items[$i]}""' >>"$tmp_json_file"
    if [[ $i -lt $((array_size - 1)) ]]; then
        echo -n ',' >>"$tmp_json_file"
    fi
done

echo -n ']]]' >>"$tmp_json_file"

local response
response=$(aws ec2 describe-instance-types --filters file://"${tmp_json_file}" \
    --query 'InstanceTypes[*].[InstanceType]' --output text)

local error_code=$?

rm "$tmp_json_file"

if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    echo "ERROR: AWS reports describe-instance-types operation failed."
    return 1
fi

echo "$response"
return 0
}

```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```
#####
```

```

# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then
        errecho " The service returned an error."
    elif [ "$err_code" == 255 ]; then
        errecho " 255 is a catch-all error."
    fi

    return 0
}

```

- API-Einheiten finden Sie [DescribeInstanceTypes](#) in der AWS CLI Befehlsreferenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie einen Instance-Typ

Im folgenden `describe-instance-types`-Beispiel werden die Details für den angegebenen Instance-Typ angezeigt.

```
aws ec2 describe-instance-types \  
  --instance-types t2.micro
```

Ausgabe:

```
{  
  "InstanceTypes": [  
    {  
      "InstanceType": "t2.micro",  
      "CurrentGeneration": true,  
      "FreeTierEligible": true,  
      "SupportedUsageClasses": [  
        "on-demand",  
        "spot"  
      ],  
      "SupportedRootDeviceTypes": [  
        "ebs"  
      ],  
      "BareMetal": false,  
      "Hypervisor": "xen",  
      "ProcessorInfo": {  
        "SupportedArchitectures": [  
          "i386",  
          "x86_64"  
        ],  
        "SustainedClockSpeedInGhz": 2.5  
      },  
      "VCpuInfo": {  
        "DefaultVCpus": 1,  
        "DefaultCores": 1,  
        "DefaultThreadsPerCore": 1,  
        "ValidCores": [  
          1  
        ],  
      },  
    },  
  ],  
}
```

```
        "ValidThreadsPerCore": [
            1
        ]
    },
    "MemoryInfo": {
        "SizeInMiB": 1024
    },
    "InstanceStorageSupported": false,
    "EbsInfo": {
        "EbsOptimizedSupport": "unsupported",
        "EncryptionSupport": "supported"
    },
    "NetworkInfo": {
        "NetworkPerformance": "Low to Moderate",
        "MaximumNetworkInterfaces": 2,
        "Ipv4AddressesPerInterface": 2,
        "Ipv6AddressesPerInterface": 2,
        "Ipv6Supported": true,
        "EnaSupport": "unsupported"
    },
    "PlacementGroupInfo": {
        "SupportedStrategies": [
            "partition",
            "spread"
        ]
    },
    "HibernationSupported": false,
    "BurstablePerformanceSupported": true,
    "DedicatedHostsSupported": false,
    "AutoRecoverySupported": true
    }
}
]
```

Weitere Informationen finden Sie unter [Instance-Typen](#) im Amazon Elastic Compute Cloud-Benutzerhandbuch für Linux-Instances.

Beispiel 2: So filtern Sie die verfügbaren Instance-Typen

Sie können einen Filter angeben, um die Ergebnisse auf Instance-Typen mit einem bestimmten Merkmal zu beschränken. Im folgenden `describe-instance-types`-Beispiel werden die Instance-Typen aufgeführt, die den Ruhezustand unterstützen.

```
aws ec2 describe-instance-types \
  --filters Name=hibernation-supported,Values=true --query
  'InstanceTypes[*].InstanceType'
```

Ausgabe:

```
[
  "m5.8xlarge",
  "r3.large",
  "c3.8xlarge",
  "r5.large",
  "m4.4xlarge",
  "c4.large",
  "m5.xlarge",
  "m4.xlarge",
  "c3.large",
  "c4.8xlarge",
  "c4.4xlarge",
  "c5.xlarge",
  "c5.12xlarge",
  "r5.4xlarge",
  "c5.4xlarge"
]
```

Weitere Informationen finden Sie unter [Instance-Typen](#) im Amazon Elastic Compute Cloud-Benutzerhandbuch für Linux-Instances.

- API-Einheiten finden Sie [DescribeInstanceTypes](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
```

```

    * Asynchronously retrieves the instance types available in the current AWS
    region.
    * <p>
    * This method uses the AWS SDK's asynchronous API to fetch the available
    instance types
    * and then processes the response. It logs the memory information, network
    information,
    * and instance type for each instance type returned. Additionally, it
    returns a
    * {@link CompletableFuture} that resolves to the instance type string for
    the "t2.2xlarge"
    * instance type, if it is found in the response. If the "t2.2xlarge"
    instance type is not
    * found, an empty string is returned.
    * </p>
    *
    * @return a {@link CompletableFuture} that resolves to the instance type
    string for the
    * "t2.2xlarge" instance type, or an empty string if the instance type is not
    found
    */
    public CompletableFuture<String> getInstanceTypesAsync() {
        DescribeInstanceTypesRequest typesRequest =
        DescribeInstanceTypesRequest.builder()
            .maxResults(10)
            .build();

        CompletableFuture<DescribeInstanceTypesResponse> response =
        getAsyncClient().describeInstanceTypes(typesRequest);
        response.whenComplete((resp, ex) -> {
            if (resp != null) {
                List<InstanceTypeInfo> instanceTypes = resp.instanceTypes();
                for (InstanceTypeInfo type : instanceTypes) {
                    logger.info("The memory information of this type is " +
                    type.memoryInfo().sizeInMiB());
                    logger.info("Network information is " +
                    type.networkInfo().toString());
                    logger.info("Instance type is " +
                    type.instanceType().toString());
                }
            } else {
                throw (RuntimeException) ex;
            }
        });
    }

```

```
return response.thenApply(resp -> {
    for (InstanceTypeInfo type : resp.instanceTypes()) {
        String instanceType = type.instanceType().toString();
        if (instanceType.equals("t2.2xlarge")) {
            return instanceType;
        }
    }
    return "";
});
}
```

- API Einzelheiten finden Sie [DescribeInstanceTypes](#) unter AWS SDK for Java 2.x API Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import { EC2Client, paginateDescribeInstanceTypes } from "@aws-sdk/client-ec2";

/**
 * Describes the specified instance types. By default, all instance types for the
 * current Region are described. Alternatively, you can filter the results.
 * @param {{ pageSize: string, supportedArch: string[], freeTier: boolean }}
 * options
 */
export const main = async ({ pageSize, supportedArch, freeTier }) => {
    pageSize = Number.parseInt(pageSize);
    const client = new EC2Client({});

    // The paginate function is a wrapper around the underlying command.
    const paginator = paginateDescribeInstanceTypes(
```

```
// Without limiting the page size, this call can take a long time. pageSize
is just sugar for
// the MaxResults property in the underlying command.
{ client, pageSize },
{
  Filters: [
    {
      Name: "processor-info.supported-architecture",
      Values: supportedArch,
    },
    { Name: "free-tier-eligible", Values: [freeTier ? "true" : "false"] },
  ],
},
);

try {
  /**
   * @type {import('@aws-sdk/client-ec2').InstanceTypeInfo[]}
   */
  const instanceTypes = [];

  for await (const page of paginator) {
    if (page.InstanceTypes.length) {
      instanceTypes.push(...page.InstanceTypes);

      // When we have at least 1 result, we can stop.
      if (instanceTypes.length >= 1) {
        break;
      }
    }
  }
  console.log(
    `Memory size in MiB for matching instance types:\n\n
    ${instanceTypes.map((it) => `${it.InstanceType}: ${it.MemoryInfo.SizeInMiB}
    MiB`).join("\n")}`
  );
} catch (caught) {
  if (caught instanceof Error && caught.name === "InvalidParameterValue") {
    console.warn(`${caught.message}`);
    return [];
  }
  throw caught;
}
};
```

- API-Einheiten finden Sie [DescribeInstanceTypes](#) unter AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
// Get a list of instance types.
suspend fun getInstanceTypesSc(): String {
    var instanceType = ""
    val filterObs = ArrayList<Filter>()
    val filter =
        Filter {
            name = "processor-info.supported-architecture"
            values = listOf("arm64")
        }

    filterObs.add(filter)
    val typesRequest =
        DescribeInstanceTypesRequest {
            filters = filterObs
            maxResults = 10
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeInstanceTypes(typesRequest)
        response.instanceTypes?.forEach { type ->
            println("The memory information of this type is
${type.memoryInfo?.sizeInMib}")
            println("Maximum number of network cards is
${type.networkInfo?.maximumNetworkCards}")
            instanceType = type.instanceType.toString()
        }
    }
    return instanceType
}
```

```

    }
}

```

- API-Einheiten finden Sie [DescribeInstanceTypes](#) in der AWS SDK-API-Kotlin-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
        objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

```



```

        :return: An instance of EC2InstanceWrapper initialized with the default
        EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def get_instance_types(
        self, architecture: str = "x86_64", sizes: List[str] = ["*.micro",
        "*.small"]
    ) -> List[Dict[str, Any]]:
        """
        Gets instance types that support the specified architecture and size.
        See https://docs.aws.amazon.com/AWSEC2/latest/APIReference/
        API\_DescribeInstanceTypes.html
        for a list of allowable parameters.

        :param architecture: The architecture supported by instance types.
        Default: 'x86_64'.
        :param sizes: The size of instance types. Default: '*.micro', '*.small',
        :return: A list of dictionaries representing instance types that support
        the specified architecture and size.
        """
        try:
            inst_types = []
            paginator = self.ec2_client.get_paginator("describe_instance_types")
            for page in paginator.paginate(
                Filters=[
                    {
                        "Name": "processor-info.supported-architecture",
                        "Values": [architecture],
                    },
                    {"Name": "instance-type", "Values": sizes},
                ]
            ):
                inst_types += page["InstanceTypes"]
        except ClientError as err:
            logger.error(
                f"Failed to get instance types: {architecture},
                {'.'.join(map(str, sizes))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidParameterValue":
                logger.error(

```

```

        "Parameters are invalid. "
        "Ensure architecture and size strings conform to
DescribeInstanceTypes API reference."
    )
    raise
else:
    return inst_types

```

- API-Einheiten finden Sie unter [DescribeInstanceTypes](#) Python (Boto3) API -Referenz. AWS SDK

Rust

SDK für Rust

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

/// List instance types that match an image's architecture and are free tier
eligible.
pub async fn list_instance_types(&self, image: &Image) ->
Result<Vec<InstanceType>, EC2Error> {
    let architecture = format!(
        "{}",
        image.architecture().ok_or_else(|| EC2Error::new(format!(
            "Image {:?} does not have a listed architecture",
            image.image_id()
        )))?
    );
    let free_tier_eligible_filter = Filter::builder()
        .name("free-tier-eligible")
        .values("false")
        .build();
    let supported_architecture_filter = Filter::builder()
        .name("processor-info.supported-architecture")
        .values(architecture)

```

```
        .build();
    let response = self
        .client
        .describe_instance_types()
        .filters(free_tier_eligible_filter)
        .filters(supported_architecture_filter)
        .send()
        .await?;

    Ok(response
        .instance_types
        .unwrap_or_default()
        .into_iter()
        .filter_map(|iti| iti.instance_type)
        .collect())
}
```

- API-Einheiten finden Sie [DescribeInstanceTypes](#) in der AWS SDK API Rust-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeInstances** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DescribeInstances`.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)
- [Erstellen und Verwalten eines ausfallsicheren Services](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Get information about EC2 instances with a particular state.
/// </summary>
/// <param name="tagName">The name of the tag to filter on.</param>
/// <param name="tagValue">The value of the tag to look for.</param>
/// <returns>True if successful.</returns>
public async Task<bool> GetInstancesWithState(string state)
{
    try
    {
        // Filters the results of the instance list.
        var filters = new List<Filter>
        {
            new Filter
            {
                Name = $"instance-state-name",
                Values = new List<string> { state, },
            },
        };
        var request = new DescribeInstancesRequest { Filters = filters, };

        Console.WriteLine($"\\nShowing instances with state {state}");
        var paginator = _amazonEC2.Paginators.DescribeInstances(request);

        await foreach (var response in paginator.Responses)
        {
            foreach (var reservation in response.Reservations)
            {
                foreach (var instance in reservation.Instances)
                {
                    Console.WriteLine($"Instance ID: {instance.InstanceId} ");
                }
            }
        }
    }
}
```

```

        Console.WriteLine($"{\tCurrent State:
{instance.State.Name}");
    }
}

return true;
}
catch (AmazonEC2Exception ec2Exception)
{
    if (ec2Exception.ErrorCode == "InvalidParameterValue")
    {
        _logger.LogError(
            $"Invalid parameter value for filtering instances.");
    }

    return false;
}
catch (Exception ex)
{
    Console.WriteLine($"Couldn't list instances because: {ex.Message}");
    return false;
}
}

```

- API-Einheiten finden Sie [DescribeInstances](#) unter AWS SDK for .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

#####
# function ec2_describe_instances
#

```

```
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
instances.
#
# Parameters:
#     -i instance_id - The ID of the instance to describe (optional).
#     -q query - The query to filter the response (optional).
#     -h - Display help.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_describe_instances() {
    local instance_id query response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_describe_instances"
        echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
instances."
        echo "  -i instance_id - The ID of the instance to describe (optional)."
        echo "  -q query - The query to filter the response (optional)."
        echo "  -h - Display help."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "i:q:h" option; do
        case "${option}" in
            i) instance_id="${OPTARG}" ;;
            q) query="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
    export OPTIND=1
```

```

local aws_cli_args=()

if [[ -n "$instance_id" ]]; then
    # shellcheck disable=SC2206
    aws_cli_args+=("--instance-ids" $instance_id)
fi

local query_arg=""
if [[ -n "$query" ]]; then
    query_arg="--query '$query'"
else
    query_arg="--query Reservations[*].Instances[*].
[InstanceId,ImageId,InstanceType,KeyName,VpcId,PublicIpAddress,State.Name]"
fi

# shellcheck disable=SC2086
response=$(aws ec2 describe-instances \
    "${aws_cli_args[@]}" \
    $query_arg \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-instances operation failed.$response"
    return 1
}

echo "$response"

return 0
}

```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}


```

```
#####  
# function aws_cli_error_log()  
#  
# This function is used to log the error messages from the AWS CLI.  
#  
# The function expects the following argument:  
#     $1 - The error code returned by the AWS CLI.  
#  
# Returns:  
#     0: - Success.  
#  
#####  
function aws_cli_error_log() {  
    local err_code=$1  
    errecho "Error code : $err_code"  
    if [ "$err_code" == 1 ]; then  
        errecho " One or more S3 transfers failed."  
    elif [ "$err_code" == 2 ]; then  
        errecho " Command line failed to parse."  
    elif [ "$err_code" == 130 ]; then  
        errecho " Process received SIGINT."  
    elif [ "$err_code" == 252 ]; then  
        errecho " Command syntax invalid."  
    elif [ "$err_code" == 253 ]; then  
        errecho " The system environment or configuration was invalid."  
    elif [ "$err_code" == 254 ]; then  
        errecho " The service returned an error."  
    elif [ "$err_code" == 255 ]; then  
        errecho " 255 is a catch-all error."  
    fi  
  
    return 0  
}
```

- API-Einheiten finden Sie [DescribeInstances](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#!/ Describe all Amazon Elastic Compute Cloud (Amazon EC2) instances associated
with an account.
/*!
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeInstances(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeInstancesRequest request;
    bool header = false;
    bool done = false;
    while (!done) {
        Aws::EC2::Model::DescribeInstancesOutcome outcome =
ec2Client.DescribeInstances(request);
        if (outcome.IsSuccess()) {
            if (!header) {
                std::cout << std::left <<
                    std::setw(48) << "Name" <<
                    std::setw(20) << "ID" <<
                    std::setw(25) << "Ami" <<
                    std::setw(15) << "Type" <<
                    std::setw(15) << "State" <<
                    std::setw(15) << "Monitoring" << std::endl;
                header = true;
            }

            const std::vector<Aws::EC2::Model::Reservation> &reservations =
                outcome.GetResult().GetReservations();

            for (const auto &reservation: reservations) {
                const std::vector<Aws::EC2::Model::Instance> &instances =
```

```

        reservation.GetInstances();
        for (const auto &instance: instances) {
            Aws::String instanceStateString =

Aws::EC2::Model::InstanceStateNameMapper::GetNameForInstanceStateName(
                instance.GetState().GetName());

            Aws::String typeString =

Aws::EC2::Model::InstanceTypeMapper::GetNameForInstanceType(
                instance.GetInstanceType());

            Aws::String monitorString =

Aws::EC2::Model::MonitoringStateMapper::GetNameForMonitoringState(
                instance.GetMonitoring().GetState());
            Aws::String name = "Unknown";

            const std::vector<Aws::EC2::Model::Tag> &tags =
instance.GetTags();
            auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
                [](const Aws::EC2::Model::Tag
&tag) {
                    return tag.GetKey() ==

"Name";
                });
            if (nameIter != tags.cend()) {
                name = nameIter->GetValue();
            }
            std::cout <<
                std::setw(48) << name <<
                std::setw(20) << instance.GetInstanceId() <<
                std::setw(25) << instance.GetImageId() <<
                std::setw(15) << typeString <<
                std::setw(15) << instanceStateString <<
                std::setw(15) << monitorString << std::endl;
        }
    }

    if (!outcome.GetResult().GetNextToken().empty()) {
        request.SetNextToken(outcome.GetResult().GetNextToken());
    } else {
        done = true;
    }
}

```

```
    } else {
        std::cerr << "Failed to describe EC2 instances:" <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

return true;
}
```

- API-Einheiten finden Sie [DescribeInstances](#) unter AWS SDK for C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie eine Instance

Das folgende `describe-instances`-Beispiel beschreibt die angegebene Instance.

```
aws ec2 describe-instances \
  --instance-ids i-1234567890abcdef0
```

Ausgabe:

```
{
  "Reservations": [
    {
      "Groups": [],
      "Instances": [
        {
          "AmiLaunchIndex": 0,
          "ImageId": "ami-0abcdef1234567890",
          "InstanceId": "i-1234567890abcdef0",
          "InstanceType": "t3.nano",
          "KeyName": "my-key-pair",
          "LaunchTime": "2022-11-15T10:48:59+00:00",
          "Monitoring": {
            "State": "disabled"
          },
          "Placement": {
            "AvailabilityZone": "us-east-2a",
```

```
        "GroupName": "",
        "Tenancy": "default"
    },
    "PrivateDnsName": "ip-10-0-0-157.us-east-2.compute.internal",
    "PrivateIpAddress": "10-0-0-157",
    "ProductCodes": [],
    "PublicDnsName": "ec2-34-253-223-13.us-
east-2.compute.amazonaws.com",
    "PublicIpAddress": "34.253.223.13",
    "State": {
        "Code": 16,
        "Name": "running"
    },
    "StateTransitionReason": "",
    "SubnetId": "subnet-04a636d18e83cfac",
    "VpcId": "vpc-1234567890abcdef0",
    "Architecture": "x86_64",
    "BlockDeviceMappings": [
        {
            "DeviceName": "/dev/xvda",
            "Ebs": {
                "AttachTime": "2022-11-15T10:49:00+00:00",
                "DeleteOnTermination": true,
                "Status": "attached",
                "VolumeId": "vol-02e6ccdca7de29cf2"
            }
        }
    ],
    "ClientToken": "1234abcd-1234-abcd-1234-d46a8903e9bc",
    "EbsOptimized": true,
    "EnaSupport": true,
    "Hypervisor": "xen",
    "IamInstanceProfile": {
        "Arn": "arn:aws:iam::111111111111:instance-profile/
AmazonSSMRoleForInstancesQuickSetup",
        "Id": "11111111111111111111"
    },
    "NetworkInterfaces": [
        {
            "Association": {
                "IpOwnerId": "amazon",
                "PublicDnsName": "ec2-34-253-223-13.us-
east-2.compute.amazonaws.com",
                "PublicIp": "34.253.223.13"
```

```

    },
    "Attachment": {
      "AttachTime": "2022-11-15T10:48:59+00:00",
      "AttachmentId": "eni-attach-1234567890abcdefg",
      "DeleteOnTermination": true,
      "DeviceIndex": 0,
      "Status": "attached",
      "NetworkCardIndex": 0
    },
    "Description": "",
    "Groups": [
      {
        "GroupName": "launch-wizard-146",
        "GroupId": "sg-1234567890abcdefg"
      }
    ],
    "Ipv6Addresses": [],
    "MacAddress": "00:11:22:33:44:55",
    "NetworkInterfaceId": "eni-1234567890abcdefg",
    "OwnerId": "104024344472",
    "PrivateDnsName": "ip-10-0-0-157.us-
east-2.compute.internal",
    "PrivateIpAddress": "10-0-0-157",
    "PrivateIpAddresses": [
      {
        "Association": {
          "IpOwnerId": "amazon",
          "PublicDnsName": "ec2-34-253-223-13.us-
east-2.compute.amazonaws.com",
          "PublicIp": "34.253.223.13"
        },
        "Primary": true,
        "PrivateDnsName": "ip-10-0-0-157.us-
east-2.compute.internal",
        "PrivateIpAddress": "10-0-0-157"
      }
    ],
    "SourceDestCheck": true,
    "Status": "in-use",
    "SubnetId": "subnet-1234567890abcdefg",
    "VpcId": "vpc-1234567890abcdefg",
    "InterfaceType": "interface"
  }
],

```

```
"RootDeviceName": "/dev/xvda",
"RootDeviceType": "ebs",
"SecurityGroups": [
  {
    "GroupName": "launch-wizard-146",
    "GroupId": "sg-1234567890abcdefg"
  }
],
"SourceDestCheck": true,
"Tags": [
  {
    "Key": "Name",
    "Value": "my-instance"
  }
],
"VirtualizationType": "hvm",
"CpuOptions": {
  "CoreCount": 1,
  "ThreadsPerCore": 2
},
"CapacityReservationSpecification": {
  "CapacityReservationPreference": "open"
},
"HibernationOptions": {
  "Configured": false
},
"MetadataOptions": {
  "State": "applied",
  "HttpTokens": "optional",
  "HttpPutResponseHopLimit": 1,
  "HttpEndpoint": "enabled",
  "HttpProtocolIpv6": "disabled",
  "InstanceMetadataTags": "enabled"
},
"EnclaveOptions": {
  "Enabled": false
},
"PlatformDetails": "Linux/UNIX",
"UsageOperation": "RunInstances",
"UsageOperationUpdateTime": "2022-11-15T10:48:59+00:00",
"PrivateDnsNameOptions": {
  "HostnameType": "ip-name",
  "EnableResourceNameDnsARecord": true,
  "EnableResourceNameDnsAAAARecord": false
}
```

```

        },
        "MaintenanceOptions": {
            "AutoRecovery": "default"
        }
    }
],
"OwnerId": "111111111111",
"ReservationId": "r-1234567890abcdefg"
}
]
}

```

Beispiel 2: So filtern Sie nach Instances mit dem angegebenen Typ

Im folgenden `describe-instances`-Beispiel werden Filter verwendet, um die Ergebnisse auf Instances des angegebenen Typs zu beschränken.

```

aws ec2 describe-instances \
  --filters Name=instance-type,Values=m5.large

```

Ein Beispiel für eine Ausgabe finden Sie in Beispiel 1.

Weitere Informationen finden Sie unter [Auflisten und Filtern mithilfe von CLI im EC2 Amazon-Benutzerhandbuch](#).

Beispiel 3: So filtern Sie nach Instances mit dem angegebenen Typ und der angegebenen Availability Zone

Im folgenden `describe-instances`-Beispiel werden mehrere Filter verwendet, um die Ergebnisse auf Instances mit dem angegebenen Typ zu beschränken, die sich ebenfalls in der angegebenen Availability Zone befinden.

```

aws ec2 describe-instances \
  --filters Name=instance-type,Values=t2.micro,t3.micro Name=availability-zone,Values=us-east-2c

```

Ein Beispiel für eine Ausgabe finden Sie in Beispiel 1.

Beispiel 4: Um mithilfe einer JSON Datei nach Instances mit dem angegebenen Typ und der angegebenen Availability Zone zu filtern

Das folgende `describe-instances` Beispiel verwendet eine JSON Eingabedatei, um dieselbe Filterung wie im vorherigen Beispiel durchzuführen. Wenn Filter komplizierter werden, können sie einfacher in einer JSON Datei angegeben werden.

```
aws ec2 describe-instances \  
  --filters file://filters.json
```

Inhalt von `filters.json`:

```
[  
  {  
    "Name": "instance-type",  
    "Values": ["t2.micro", "t3.micro"]  
  },  
  {  
    "Name": "availability-zone",  
    "Values": ["us-east-2c"]  
  }  
]
```

Ein Beispiel für eine Ausgabe finden Sie in Beispiel 1.

Beispiel 5: So filtern Sie nach Instances mit dem angegebenen Owner-Tag

Im folgenden `describe-instances`-Beispiel werden Tag-Filter verwendet, um die Ergebnisse unabhängig vom Tag-Wert auf Instances zu beschränken, die über ein Tag mit dem angegebenen Tag-Schlüssel (Owner) verfügen.

```
aws ec2 describe-instances \  
  --filters "Name=tag-key, Values=Owner"
```

Ein Beispiel für eine Ausgabe finden Sie in Beispiel 1.

Beispiel 6: So filtern Sie nach Instances mit dem angegebenen my-team-Tag-Wert

Im folgenden `describe-instances`-Beispiel werden Tag-Filter verwendet, um die Ergebnisse auf Instances zu beschränken, die ein Tag mit dem angegebenen Tag-Wert (my-team) haben, unabhängig vom Tag-Schlüssel.

```
aws ec2 describe-instances \  
  --filters "Name=tag-value, Values=my-team"
```


Ein Beispiel für eine Ausgabe finden Sie in Beispiel 1.

Beispiel 7: So filtern Sie nach Instances mit dem angegebenen Besitzer-Tag und my-team-Wert

Im folgenden `describe-instances`-Beispiel werden Tag-Filter verwendet, um die Ergebnisse auf Instances zu beschränken, die das angegebene Tag haben (Besitzer=my-team).

```
aws ec2 describe-instances \
  --filters "Name=tag:Owner,Values=my-team"
```

Ein Beispiel für eine Ausgabe finden Sie in Beispiel 1.

Beispiel 8: Um nur Instanz und Subnetz IDs für alle Instanzen anzuzeigen

In den folgenden `describe-instances` Beispielen `--query` wird der Parameter verwendet, um nur die Instanz und das Subnetz IDs für alle Instanzen im JSON Format anzuzeigen.

Linux und macOS:

```
aws ec2 describe-instances \
  --query 'Reservations[*].Instances[*].{Instance:InstanceId,Subnet:SubnetId}' \
  --output json
```

Windows:

```
aws ec2 describe-instances ^
  --query "Reservations[*].Instances[*].{Instance:InstanceId,Subnet:SubnetId}" ^
  --output json
```

Ausgabe:

```
[
  {
    "Instance": "i-057750d42936e468a",
    "Subnet": "subnet-069beee9b12030077"
  },
  {
```

```

    "Instance": "i-001efd250faaa6ffa",
    "Subnet": "subnet-0b715c6b7db68927a"
  },
  {
    "Instance": "i-027552a73f021f3bd",
    "Subnet": "subnet-0250c25a1f4e15235"
  }
  ...
]

```

Beispiel 9: Um Instanzen des angegebenen Typs zu filtern und nur ihre Instanz anzuzeigen IDs

Im folgenden `describe-instances` Beispiel werden Filter verwendet, um die Ergebnisse auf Instanzen des angegebenen Typs zu beschränken, und der `--query` Parameter verwendet, um nur die Instanz anzuzeigen IDs.

```

aws ec2 describe-instances \
  --filters "Name=instance-type,Values=t2.micro" \
  --query "Reservations[*].Instances[*].[InstanceId]" \
  --output text

```

Ausgabe:

```

i-031c0dc19de2fb70c
i-00d8bfff789a736b75
i-0b715c6b7db68927a
i-0626d4edd54f1286d
i-00b8ae04f9f99908e
i-0fc71c25d2374130c

```

Beispiel 10: Um Instances des angegebenen Typs zu filtern und nur ihre InstanzIDs, Availability Zone und den angegebenen Tag-Wert anzuzeigen

In den folgenden `describe-instances`-Beispielen werden die Instance-ID, die Availability Zone und der Wert des Name-Tags für Instances, die ein Tag mit dem Namen `tag-key` haben, im Tabellenformat angezeigt.

Linux und macOS:

```

aws ec2 describe-instances \

```

```

--filters Name=tag-key,Values=Name \
--query 'Reservations[*].Instances[*].
{Instance:InstanceId,AZ:Placement.AvailabilityZone,Name:Tags[?Key=='Name `]|
[0].Value}' \
--output table

```

Windows:

```

aws ec2 describe-instances ^
--filters Name=tag-key,Values=Name ^
--query "Reservations[*].Instances[*].
{Instance:InstanceId,AZ:Placement.AvailabilityZone,Name:Tags[?Key=='Name `]|
[0].Value}" ^
--output table

```

Ausgabe:

```

-----
|                               DescribeInstances                               |
+-----+-----+-----+
|      AZ      |      Instance      |      Name      |
+-----+-----+-----+
| us-east-2b  | i-057750d42936e468a | my-prod-server |
| us-east-2a  | i-001efd250faaa6ffa | test-server-1  |
| us-east-2a  | i-027552a73f021f3bd | test-server-2  |
+-----+-----+-----+

```

Beispiel 11: So beschreiben Sie Instances in einer Partition-Placement-Gruppe

Das folgende `describe-instances`-Beispiel beschreibt die angegebene Instance. Die Ausgabe enthält die Platzierungsinformationen für die Instance, die den Namen der Platzierungsgruppe und die Partitionsnummer für die Instance enthalten.

```

aws ec2 describe-instances \
--instance-ids i-0123a456700123456 \
--query "Reservations[*].Instances[*].Placement"

```

Ausgabe:

```

[
  [

```

```

    {
      "AvailabilityZone": "us-east-1c",
      "GroupName": "HDFS-Group-A",
      "PartitionNumber": 3,
      "Tenancy": "default"
    }
  ]
]

```

Weitere Informationen finden Sie unter [Beschreibung von Instances in einer Placement-Gruppe](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 12: So filtern Sie auf Instances mit der angegebenen Platzierungsgruppe und Partitionsnummer

Im folgenden `describe-instances`-Beispiel werden die Ergebnisse nur nach den Instances mit der angegebenen Platzierungsgruppe und Partitionsnummer gefiltert.

```

aws ec2 describe-instances \
  --filters "Name=placement-group-name,Values=HDFS-Group-A" "Name=placement-
partition-number,Values=7"

```

Im Folgenden werden nur die relevanten Informationen aus der Ausgabe angezeigt.

```

"Instances": [
  {
    "InstanceId": "i-0123a456700123456",
    "InstanceType": "r4.large",
    "Placement": {
      "AvailabilityZone": "us-east-1c",
      "GroupName": "HDFS-Group-A",
      "PartitionNumber": 7,
      "Tenancy": "default"
    }
  },
  {
    "InstanceId": "i-9876a543210987654",
    "InstanceType": "r4.large",
    "Placement": {
      "AvailabilityZone": "us-east-1c",
      "GroupName": "HDFS-Group-A",

```

```
        "PartitionNumber": 7,  
        "Tenancy": "default"  
    }  
],
```

Weitere Informationen finden Sie unter [Beschreibung von Instances in einer Placement-Gruppe](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 13: So filtern Sie nach Instances, die so konfiguriert sind, dass sie den Zugriff auf Tags aus Instance-Metadaten erlauben

Im folgenden `describe-instances`-Beispiel werden die Ergebnisse nur nach den Instances gefiltert, die so konfiguriert sind, dass sie den Zugriff auf Instance-Tags aus den Instance-Metadaten ermöglichen.

```
aws ec2 describe-instances \  
  --filters "Name=metadata-options.instance-metadata-tags,Values=enabled" \  
  --query "Reservations[*].Instances[*].InstanceId" \  
  --output text
```

Im Folgenden wird die erwartete Ausgabe dargestellt.

```
i-1234567890abcdefg  
i-abcdefg1234567890  
i-111111111aaaaaaaa  
i-aaaaaaaa111111111
```

Weitere Informationen finden Sie unter [Arbeiten mit Instance-Tags in Instance-Metadaten](#) im EC2Amazon-Benutzerhandbuch.

- API-Einheiten finden Sie [DescribeInstances](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

/**
 * Asynchronously describes an AWS EC2 image with the specified image ID.
 *
 * @param imageId the ID of the image to be described
 * @return a {@link CompletableFuture} that, when completed, contains the ID
of the described image
 * @throws RuntimeException if no images are found with the provided image
ID, or if an error occurs during the AWS API call
 */
public CompletableFuture<String> describeImageAsync(String imageId) {
    DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
        .imageIds(imageId)
        .build();

    AtomicReference<String> imageIdRef = new AtomicReference<>();
    DescribeImagesPublisher paginator =
getAsyncClient().describeImagesPaginator(imagesRequest);
    return paginator.subscribe(response -> {
        response.images().stream()
            .filter(image -> image.imageId().equals(imageId))
            .findFirst()
            .ifPresent(image -> {
                logger.info("The description of the image is " +
image.description());
                logger.info("The name of the image is " + image.name());
                imageIdRef.set(image.imageId());
            });
    }).thenApply(v -> {
        String id = imageIdRef.get();
        if (id == null) {
            throw new RuntimeException("No images found with the provided
image ID.");
        }
        return id;
    }).exceptionally(ex -> {
        logger.info("Failed to describe image: " + ex.getMessage());
        throw new RuntimeException("Failed to describe image", ex);
    });
}

```

- API-Einheiten finden Sie [DescribeInstances](#) unter AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import { EC2Client, paginateDescribeInstances } from "@aws-sdk/client-ec2";

/**
 * List all of your EC2 instances running with the provided architecture that
 * were launched in the past month.
 * @param {{ pageSize: string, architectures: string[] }} options
 */
export const main = async ({ pageSize, architectures }) => {
  pageSize = Number.parseInt(pageSize);
  const client = new EC2Client({});
  const d = new Date();
  const year = d.getFullYear();
  const month = `0${d.getMonth() + 1}`.slice(-2);
  const launchTimePattern = `${year}-${month}-*`;

  const paginator = paginateDescribeInstances(
    {
      client,
      pageSize,
    },
    {
      Filters: [
        { Name: "architecture", Values: architectures },
        { Name: "instance-state-name", Values: ["running"] },
        {
          Name: "launch-time",
          Values: [launchTimePattern],
        },
      ],
    },
  );
};
```

```

try {
  /**
   * @type {import('@aws-sdk/client-ec2').Instance[]}
   */
  const instanceList = [];
  for await (const page of paginator) {
    const { Reservations } = page;
    for (const reservation of Reservations) {
      instanceList.push(...reservation.Instances);
    }
  }
  console.log(
    `Running instances launched this month:\n\n${instanceList.map((instance) =>
instance.InstanceId).join("\n")}` ,
  );
} catch (caught) {
  if (caught instanceof Error && caught.name === "InvalidParameterValue") {
    console.warn(`${caught.message}.`);
  } else {
    throw caught;
  }
}
};

```

- API-Einheiten finden Sie [DescribeInstances](#) unter AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

suspend fun describeEC2Instances() {
  val request =
    DescribeInstancesRequest {
      maxResults = 6
    }
}

```



```

Ec2Client { region = "us-west-2" }.use { ec2 ->
    val response = ec2.describeInstances(request)
    response.reservations?.forEach { reservation ->
        reservation.instances?.forEach { instance ->
            println("Instance Id is ${instance.instanceId}")
            println("Image id is ${instance.imageId}")
            println("Instance type is ${instance.instanceType}")
            println("Instance state name is ${instance.state?.name}")
            println("monitoring information is
${instance.monitoring?.state}")
        }
    }
}
}

```

- API-Einheiten finden Sie [DescribeInstances](#) in der AWS SDK-API-Kotlin-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Instanz.

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

Ausgabe:

```

AmiLaunchIndex      : 0
Architecture        : x86_64
BlockDeviceMappings : {/dev/sda1}
ClientToken         : T1eEy1448154045270
EbsOptimized        : False
Hypervisor          : xen
IamInstanceProfile  : Amazon.EC2.Model.IamInstanceProfile
ImageId             : ami-12345678
InstanceId          : i-12345678
InstanceLifecycle   :
InstanceType        : t2.micro
KernelId            :
KeyName             : my-key-pair

```

```

LaunchTime      : 12/4/2015 4:44:40 PM
Monitoring     : Amazon.EC2.Model.Monitoring
NetworkInterfaces : {ip-10-0-2-172.us-west-2.compute.internal}
Placement      : Amazon.EC2.Model.Placement
Platform       : Windows
PrivateDnsName  : ip-10-0-2-172.us-west-2.compute.internal
PrivateIpAddress : 10.0.2.172
ProductCodes   : {}
PublicDnsName   :
PublicIpAddress :
RamdiskId      :
RootDeviceName  : /dev/sda1
RootDeviceType  : ebs
SecurityGroups  : {default}
SourceDestCheck : True
SpotInstanceRequestId :
SriovNetSupport :
State          : Amazon.EC2.Model.InstanceState
StateReason     :
StateTransitionReason :
SubnetId       : subnet-12345678
Tags           : {Name}
VirtualizationType : hvm
VpcId         : vpc-12345678

```

Beispiel 2: Dieses Beispiel beschreibt alle Ihre Instances in der aktuellen Region, gruppiert nach Reservierungen. Um die Instanzdetails zu sehen, erweitern Sie die Instanzen-Sammlung innerhalb jedes Reservierungsobjekts.

```
Get-EC2Instance
```

Ausgabe:

```

GroupNames     : {}
Groups         : {}
Instances      : {}
OwnerId        : 123456789012
RequesterId    : 226008221399
ReservationId  : r-c5df370c

GroupNames     : {}
Groups         : {}
Instances      : {}

```

```
OwnerId       : 123456789012
RequesterId   : 854251627541
ReservationId : r-63e65bab
...
```

Beispiel 3: Dieses Beispiel veranschaulicht die Verwendung eines Filters zur Abfrage von EC2 Instances in einem bestimmten Subnetz von a. VPC

```
(Get-EC2Instance -Filter @{Name="vpc-id";Values="vpc-1a2bc34d"},@{Name="subnet-id";Values="subnet-1a2b3c4d"}).Instances
```

Ausgabe:

```
InstanceId           InstanceType Platform PrivateIpAddress PublicIpAddress
SecurityGroups SubnetId           VpcId
-----
-----
i-01af...82cf180e19 t2.medium    Windows  10.0.0.98      ...
                subnet-1a2b3c4d vpc-1a2b3c4d
i-0374...7e9d5b0c45 t2.xlarge    Windows  10.0.0.53      ...
                subnet-1a2b3c4d vpc-1a2b3c4d
```

- API-Einheiten finden Sie unter [DescribeInstances AWS Tools for PowerShell](#) Cmdlet-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
```

```
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def display(self, state_filter: Optional[str] = "running") -> None:
        """
        Displays information about instances, filtering by the specified state.

        :param state_filter: The instance state to include in the output. Only
instances in this state
                           will be displayed. Default is 'running'. Example
states: 'running', 'stopped'.
        """
        if not self.instances:
            logger.info("No instances to display.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        paginator = self.ec2_client.get_paginator("describe_instances")
        page_iterator = paginator.paginate(InstanceIds=instance_ids)
```

```

try:
    for page in page_iterator:
        for reservation in page["Reservations"]:
            for instance in reservation["Instances"]:
                instance_state = instance["State"]["Name"]

                # Apply the state filter (default is 'running')
                if state_filter and instance_state != state_filter:
                    continue # Skip this instance if it doesn't match
the filter

                # Create a formatted string with instance details
                instance_info = (
                    f". ID: {instance['InstanceId']}\n"
                    f". Image ID: {instance['ImageId']}\n"
                    f". Instance type: {instance['InstanceType']}\n"
                    f". Key name: {instance['KeyName']}\n"
                    f". VPC ID: {instance['VpcId']}\n"
                    f". Public IP: {instance.get('PublicIpAddress', 'N/A')}\n"

                    f". State: {instance_state}"
                )
                print(instance_info)

except ClientError as err:
    logger.error(
        f"Failed to display instance(s). : {' '.join(map(str,
instance_ids))}"
    )
    error_code = err.response["Error"]["Code"]
    if error_code == "InvalidInstanceID.NotFound":
        logger.error(
            "One or more instance IDs do not exist. "
            "Please verify the instance IDs and try again."
        )
        raise

```

- API-Einheiten finden Sie unter [DescribeInstances](#) Python (Boto3) API-Referenz. AWS SDK

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
require 'aws-sdk-ec2'

# @param ec2_resource [Aws::EC2::Resource] An initialized EC2 resource object.
# @example
# list_instance_ids_states(Aws::EC2::Resource.new(region: 'us-west-2'))
def list_instance_ids_states(ec2_resource)
  response = ec2_resource.instances
  if response.count.zero?
    puts 'No instances found.'
  else
    puts 'Instances -- ID, state:'
    response.each do |instance|
      puts "#{instance.id}, #{instance.state.name}"
    end
  end
end

rescue StandardError => e
  puts "Error getting information about instances: #{e.message}"
end

# Example usage:
def run_me
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage: ruby ec2-ruby-example-get-all-instance-info.rb REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-get-all-instance-info.rb us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
```

```

    region = 'us-west-2'
    # Otherwise, use the values as specified at the command prompt.
    else
      region = ARGV[0]
    end
    ec2_resource = Aws::EC2::Resource.new(region: region)
    list_instance_ids_states(ec2_resource)
  end

  run_me if $PROGRAM_NAME == __FILE__

```

- API-Einheiten finden Sie [DescribeInstances](#) unter AWS SDK for Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Rufen Sie Details für eine EC2 Instanz ab.

```

pub async fn describe_instance(&self, instance_id: &str) -> Result<Instance,
EC2Error> {
    let response = self
        .client
        .describe_instances()
        .instance_ids(instance_id)
        .send()
        .await?;

    let instance = response
        .reservations()
        .first()
        .ok_or_else(|| EC2Error::new(format!("No instance reservations for
{instance_id}")))?
        .instances()
        .first()

```

```

        .ok_or_else(|| {
            EC2Error::new(format!("No instances in reservation for
{instance_id}"))
        })?;

        Ok(instance.clone())
    }

```

Rufen Sie nach dem Erstellen einer EC2 Instanz deren Details ab und speichern Sie sie.

```

/// Create an EC2 instance with the given ID on a given type, using a
/// generated KeyPair and applying a list of security groups.
pub async fn create(
    &mut self,
    ec2: &EC2,
    image_id: &str,
    instance_type: InstanceType,
    key_pair: &KeyPairInfo,
    security_groups: Vec<&SecurityGroup>,
) -> Result<(), EC2Error> {
    let instance_id = ec2
        .create_instance(image_id, instance_type, key_pair, security_groups)
        .await?;
    let instance = ec2.describe_instance(&instance_id).await?;
    self.instance = Some(instance);
    Ok(())
}

```

- API-Einheiten finden Sie [DescribeInstances](#) in der AWS SDK API Rust-Referenz.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.


```

TRY.
    oo_result = lo_ec2->describeinstances( ) .
oo_result is returned for testing purposes. "

" Retrieving details of EC2 instances. "
DATA: lv_instance_id    TYPE /aws1/ec2string,
      lv_status         TYPE /aws1/ec2instancename,
      lv_instance_type  TYPE /aws1/ec2instancetype,
      lv_image_id       TYPE /aws1/ec2string.
LOOP AT oo_result->get_reservations( ) INTO DATA(lo_reservation).
  LOOP AT lo_reservation->get_instances( ) INTO DATA(lo_instance).
    lv_instance_id = lo_instance->get_instanceid( ).
    lv_status = lo_instance->get_state( )->get_name( ).
    lv_instance_type = lo_instance->get_instancetype( ).
    lv_image_id = lo_instance->get_imageid( ).
  ENDLLOOP.
ENDLOOP.
MESSAGE 'Retrieved information about EC2 instances.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
  DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
  MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- API-Einheiten finden Sie [DescribeInstances](#) unter AWS SDKSAPABAPI als Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeInternetGateways** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeInternetGateways`.

CLI

AWS CLI

Um ein Internet-Gateway zu beschreiben

Das folgende `describe-internet-gateways` Beispiel beschreibt das angegebene Internet-Gateway.

```
aws ec2 describe-internet-gateways \  
  --internet-gateway-ids igw-0d0fb496b3EXAMPLE
```

Ausgabe:

```
{  
  "InternetGateways": [  
    {  
      "Attachments": [  
        {  
          "State": "available",  
          "VpcId": "vpc-0a60eb65b4EXAMPLE"  
        }  
      ],  
      "InternetGatewayId": "igw-0d0fb496b3EXAMPLE",  
      "OwnerId": "123456789012",  
      "Tags": [  
        {  
          "Key": "Name",  
          "Value": "my-igw"  
        }  
      ]  
    }  
  ]  
}
```

Weitere Informationen finden Sie unter [Internet-Gateways](#) im VPCAmazon-Benutzerhandbuch.

- API-Einheiten finden Sie [DescribeInternetGateways](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das angegebene Internet-Gateway.

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

Ausgabe:

Attachments	InternetGatewayId	Tags
-----	-----	----
{vpc-1a2b3c4d}	igw-1a2b3c4d	{}

Beispiel 2: Dieses Beispiel beschreibt alle Ihre Internet-Gateways.

```
Get-EC2InternetGateway
```

Ausgabe:

Attachments	InternetGatewayId	Tags
-----	-----	----
{vpc-1a2b3c4d}	igw-1a2b3c4d	{}
{}	igw-2a3b4c5d	{}

- API-Einheiten finden Sie unter [DescribeInternetGateways AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeKeyPairs** mit einem AWS SDK oder CLI.

Die folgenden Codebeispiele zeigen, wie man es benutzt `DescribeKeyPairs`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Get information about an Amazon EC2 key pair.
/// </summary>
/// <param name="keyPairName">The name of the key pair.</param>
/// <returns>A list of key pair information.</returns>
public async Task<List<KeyPairInfo>> DescribeKeyPairs(string keyPairName)
{
    try
    {
        var request = new DescribeKeyPairsRequest();
        if (!string.IsNullOrEmpty(keyPairName))
        {
            request = new DescribeKeyPairsRequest
            {
                KeyNames = new List<string> { keyPairName }
            };
        }


        var response = await _amazonEC2.DescribeKeyPairsAsync(request);
        return response.KeyPairs.ToList();
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
        {
            _logger.LogError(
                $"A key pair called {keyPairName} does not exist.");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(
            $"An error occurred while describing the key pair.:
{ex.Message}");
        throw;
    }
}
```

- API-Einheiten finden Sie [DescribeKeyPairs](#) unter AWS SDK for .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

 Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#####
# function ec2_describe_key_pairs
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
# key pairs.
#
# Parameters:
#     -h - Display help.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_describe_key_pairs() {
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_describe_key_pairs"
        echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2) key
pairs."
        echo "  -h - Display help."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "h" option; do
        case "${option}" in
            h)
                usage
                return 0
            ;;
        esac
    done
}
```

```

    \?)
    echo "Invalid parameter"
    usage
    return 1
    ;;
esac
done
export OPTIND=1

local response

response=$(aws ec2 describe-key-pairs \
  --query 'KeyPairs[*].[KeyName, KeyFingerprint]' \
  --output text) || {
  aws_cli_error_log ${?}
  errecho "ERROR: AWS reports describe-key-pairs operation failed.$response"
  return 1
}

echo "$response"

return 0
}

```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.

```

```

#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then
        errecho " The service returned an error."
    elif [ "$err_code" == 255 ]; then
        errecho " 255 is a catch-all error."
    fi

    return 0
}

```

- API-Einheiten finden Sie [DescribeKeyPairs](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

/*! Describe all Amazon Elastic Compute Cloud (Amazon EC2) instance key pairs.
/*!

```

```
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::EC2::describeKeyPairs(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeKeyPairsRequest request;

    Aws::EC2::Model::DescribeKeyPairsOutcome outcome =
ec2Client.DescribeKeyPairs(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
            std::setw(32) << "Name" <<
            std::setw(64) << "Fingerprint" << std::endl;

        const std::vector<Aws::EC2::Model::KeyPairInfo> &key_pairs =
            outcome.GetResult().GetKeyPairs();
        for (const auto &key_pair: key_pairs) {
            std::cout << std::left <<
                std::setw(32) << key_pair.GetKeyName() <<
                std::setw(64) << key_pair.GetKeyFingerprint() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe key pairs:" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Einheiten finden Sie [DescribeKeyPairs](#) unter AWS SDK for C++ API-Referenz.

CLI

AWS CLI

So zeigen Sie ein Schlüsselpaar an

Im folgenden `describe-key-pairs`-Beispiel werden Informationen zu dem angegebenen Schlüsselpaar angezeigt.

```
aws ec2 describe-key-pairs \
```



```
--key-names my-key-pair
```

Ausgabe:

```
{
  "KeyPairs": [
    {
      "KeyId": "key-0b94643da6EXAMPLE",
      "KeyFingerprint":
"1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f",
      "KeyName": "my-key-pair",
      "KeyType": "rsa",
      "Tags": [],
      "CreateTime": "2022-05-27T21:51:16.000Z"
    }
  ]
}
```

Weitere Informationen finden Sie unter [Describe public keys](#) im EC2Amazon-Benutzerhandbuch.

- API-Einheiten finden Sie [DescribeKeyPairs](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Asynchronously describes the key pairs associated with the current AWS
 * account.
 *
 * @return a {@link CompletableFuture} containing the {@link
 * DescribeKeyPairsResponse} object, which provides
 * information about the key pairs.
 */
```

```

public CompletableFuture<DescribeKeyPairsResponse> describeKeysAsync() {
    CompletableFuture<DescribeKeyPairsResponse> responseFuture =
getAsyncClient().describeKeyPairs();
    responseFuture.whenComplete((response, exception) -> {
        if (exception != null) {
            throw new RuntimeException("Failed to describe key pairs: " +
exception.getMessage(), exception);
        }
    });

    return responseFuture;
}

```

- API-Einheiten finden Sie [DescribeKeyPairs](#) unter AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import { DescribeKeyPairsCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * List all key pairs in the current AWS account.
 * @param {{ dryRun: boolean }}
 */
export const main = async ({ dryRun }) => {
    const client = new EC2Client({});
    const command = new DescribeKeyPairsCommand({ DryRun: dryRun });

    try {
        const { KeyPairs } = await client.send(command);
        const keyPairList = KeyPairs.map(
            (kp) => ` • ${kp.KeyPairId}: ${kp.KeyName}`,
        ).join("\n");
        console.log("The following key pairs were found in your account:");
    }
}

```

```

    console.log(keyPairList);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "DryRunOperation") {
      console.log(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};

```

- API-Einheiten finden Sie [DescribeKeyPairs](#) unter AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

suspend fun describeEC2Keys() {
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.describeKeyPairs(DescribeKeyPairsRequest {})
        response.keyPairs?.forEach { keyPair ->
            println("Found key pair with name ${keyPair.keyName} and fingerprint
            ${ keyPair.keyFingerprint}")
        }
    }
}

```

- API-Einheiten finden Sie [DescribeKeyPairs](#) in der AWS SDK API Kotlin-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das angegebene key pair.

```
Get-EC2KeyPair -KeyName my-key-pair
```

Ausgabe:

```
KeyFingerprint                               KeyName
-----
1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca:9f:f5:f1:6f my-key-pair
```

Beispiel 2: Dieses Beispiel beschreibt alle Ihre Schlüsselpaare.

```
Get-EC2KeyPair
```

- API-Einheiten finden Sie unter [DescribeKeyPairs AWS Tools for PowerShell](#) Cmdlet-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
        directory,
```

```

        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is
stored.
                           This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                           This is a high-level object that wraps key pair actions.
Optional.
    """
    self.ec2_client = ec2_client
    self.key_pair = key_pair
    self.key_file_path: Optional[str] = None
    self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2
client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())

    def list(self, limit: Optional[int] = None) -> None:
        """
        Displays a list of key pairs for the current account.

        WARNING: Results are not paginated.

        :param limit: The maximum number of key pairs to list. If not specified,
all key pairs will be listed.
        :raises ClientError: If there is an error in listing the key pairs.
        """
        try:
            response = self.ec2_client.describe_key_pairs()
            key_pairs = response.get("KeyPairs", [])

```

```
        if limit:
            key_pairs = key_pairs[:limit]

        for key_pair in key_pairs:
            logger.info(
                f"Found {key_pair['KeyType']} key '{key_pair['KeyName']}'
with fingerprint:"
            )
            logger.info(f"\t{key_pair['KeyFingerprint']}")
        except ClientError as err:
            logger.error(f"Failed to list key pairs: {str(err)}")
            raise
```

- API-Einheiten finden Sie unter [DescribeKeyPairs](#) Python (Boto3) API-Referenz. AWS SDK

Rust

SDK für Rust

Note


Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
pub async fn list_key_pair(&self) -> Result<Vec<KeyPairInfo>, EC2Error> {
    let output = self.client.describe_key_pairs().send().await?;
    Ok(output.key_pairs.unwrap_or_default())
}
```

- API-Einheiten finden Sie [DescribeKeyPairs](#) in der AWS SDK API Rust-Referenz.

SAP ABAP

SDK für SAP ABAP

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

TRY.
    oo_result = lo_ec2->describekeypairs( ) .
    oo_result is returned for testing purposes. "
    DATA(lt_key_pairs) = oo_result->get_keypairs( ).
    MESSAGE 'Retrieved information about key pairs.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- API Einzelheiten finden Sie [DescribeKeyPairs](#) unter AWS SDKSAPABAPAPI als Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeNetworkAcls** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeNetworkAcls`.

CLI

AWS CLI

Um Ihr Netzwerk zu beschreiben ACLs

Im folgenden `describe-network-acls` Beispiel werden Details zu Ihrem Netzwerk ACLs abgerufen.

aws ec2 describe-network-acls

Ausgabe:

```
{
  "NetworkAcls": [
    {
      "Associations": [
        {
          "NetworkAclAssociationId": "aclassoc-0c1679dc41EXAMPLE",
          "NetworkAclId": "acl-0ea1f54ca7EXAMPLE",
          "SubnetId": "subnet-0931fc2fa5EXAMPLE"
        }
      ],
      "Entries": [
        {
          "CidrBlock": "0.0.0.0/0",
          "Egress": true,
          "Protocol": "-1",
          "RuleAction": "allow",
          "RuleNumber": 100
        },
        {
          "CidrBlock": "0.0.0.0/0",
          "Egress": true,
          "Protocol": "-1",
          "RuleAction": "deny",
          "RuleNumber": 32767
        },
        {
          "CidrBlock": "0.0.0.0/0",
          "Egress": false,
          "Protocol": "-1",
          "RuleAction": "allow",
          "RuleNumber": 100
        },
        {
          "CidrBlock": "0.0.0.0/0",
          "Egress": false,
          "Protocol": "-1",
          "RuleAction": "deny",
          "RuleNumber": 32767
        }
      ]
    }
  ]
}
```



```
    ],
    "IsDefault": true,
    "NetworkAclId": "acl-0ea1f54ca7EXAMPLE",
    "Tags": [],
    "VpcId": "vpc-06e4ab6c6cEXAMPLE",
    "OwnerId": "111122223333"
  },
  {
    "Associations": [],
    "Entries": [
      {
        "CidrBlock": "0.0.0.0/0",
        "Egress": true,
        "Protocol": "-1",
        "RuleAction": "allow",
        "RuleNumber": 100
      },
      {
        "Egress": true,
        "Ipv6CidrBlock": ":::/0",
        "Protocol": "-1",
        "RuleAction": "allow",
        "RuleNumber": 101
      },
      {
        "CidrBlock": "0.0.0.0/0",
        "Egress": true,
        "Protocol": "-1",
        "RuleAction": "deny",
        "RuleNumber": 32767
      },
      {
        "Egress": true,
        "Ipv6CidrBlock": ":::/0",
        "Protocol": "-1",
        "RuleAction": "deny",
        "RuleNumber": 32768
      },
      {
        "CidrBlock": "0.0.0.0/0",
        "Egress": false,
        "Protocol": "-1",
        "RuleAction": "allow",
        "RuleNumber": 100
      }
    ]
  }
}
```

```
    },
    {
      "Egress": false,
      "Ipv6CidrBlock": ":::/0",
      "Protocol": "-1",
      "RuleAction": "allow",
      "RuleNumber": 101
    },
    {
      "CidrBlock": "0.0.0.0/0",
      "Egress": false,
      "Protocol": "-1",
      "RuleAction": "deny",
      "RuleNumber": 32767
    },
    {
      "Egress": false,
      "Ipv6CidrBlock": ":::/0",
      "Protocol": "-1",
      "RuleAction": "deny",
      "RuleNumber": 32768
    }
  ],
  "IsDefault": true,
  "NetworkAclId": "acl-0e2a78e4e2EXAMPLE",
  "Tags": [],
  "VpcId": "vpc-03914afb3eEXAMPLE",
  "OwnerId": "111122223333"
}
]
```

Weitere Informationen finden Sie ACLs im AWS VPCBenutzerhandbuch unter [Netzwerk](#).

- APIEinzelheiten finden Sie [DescribeNetworkAcls](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das angegebene NetzwerkACL.

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

Ausgabe:

```
Associations : {aclassoc-1a2b3c4d}
Entries      : {Amazon.EC2.Model.NetworkAclEntry,
               Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {Name}
VpcId        : vpc-12345678
```

Beispiel 2: Dieses Beispiel beschreibt die Regeln für das angegebene NetzwerkACL.

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

Ausgabe:

```
CidrBlock    : 0.0.0.0/0
Egress       : True
IcmpTypeCode :
PortRange    :
Protocol     : -1
RuleAction   : deny
RuleNumber   : 32767

CidrBlock    : 0.0.0.0/0
Egress       : False
IcmpTypeCode :
PortRange    :
Protocol     : -1
RuleAction   : deny
RuleNumber   : 32767
```

Beispiel 3: Dieses Beispiel beschreibt Ihr gesamtes NetzwerkACLs.

```
Get-EC2NetworkAcl
```

- API-Einheiten finden Sie unter [DescribeNetworkAcls AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeNetworkInterfaceAttribute** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeNetworkInterfaceAttribute`.

CLI

AWS CLI

Um das Attachment-Attribut einer Netzwerkschnittstelle zu beschreiben

Dieser Beispielbefehl beschreibt das `attachment` Attribut der angegebenen Netzwerkschnittstelle.

Befehl:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
--attribute attachment
```

Ausgabe:

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "Attachment": {
    "Status": "attached",
    "DeviceIndex": 0,
    "AttachTime": "2015-05-21T20:02:20.000Z",
    "InstanceId": "i-1234567890abcdef0",
    "DeleteOnTermination": true,
    "AttachmentId": "eni-attach-43348162",
    "InstanceOwnerId": "123456789012"
  }
}
```

Um das Beschreibungsattribut einer Netzwerkschnittstelle zu beschreiben

Dieser Beispielbefehl beschreibt das `description` Attribut der angegebenen Netzwerkschnittstelle.

Befehl:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
--attribute description
```

Ausgabe:

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "Description": {
    "Value": "My description"
  }
}
```

Um das groupSet Attribut einer Netzwerkschnittstelle zu beschreiben

Dieser Beispielbefehl beschreibt das groupSet Attribut der angegebenen Netzwerkschnittstelle.

Befehl:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
--attribute groupSet
```

Ausgabe:

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "Groups": [
    {
      "GroupName": "my-security-group",
      "GroupId": "sg-903004f8"
    }
  ]
}
```

Um das sourceDestCheck Attribut einer Netzwerkschnittstelle zu beschreiben

Dieser Beispielbefehl beschreibt das sourceDestCheck Attribut der angegebenen Netzwerkschnittstelle.

Befehl:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
--attribute sourceDestCheck
```

Ausgabe:

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "SourceDestCheck": {
    "Value": true
  }
}
```

- API-Einheiten finden Sie [DescribeNetworkInterfaceAttribute](#) in der AWS CLI Befehlsreferenz.

PowerShell**Tools für PowerShell**

Beispiel 1: Dieses Beispiel beschreibt die angegebene Netzwerkschnittstelle.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
Attachment
```

Ausgabe:

```
Attachment          : Amazon.EC2.Model.NetworkInterfaceAttachment
```

Beispiel 2: Dieses Beispiel beschreibt die angegebene Netzwerkschnittstelle.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
Description
```

Ausgabe:

```
Description          : My description
```

Beispiel 3: Dieses Beispiel beschreibt die angegebene Netzwerkschnittstelle.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute GroupSet
```

Ausgabe:

```
Groups           : {my-security-group}
```

Beispiel 4: Dieses Beispiel beschreibt die angegebene Netzwerkschnittstelle.

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute SourceDestCheck
```

Ausgabe:

```
SourceDestCheck : True
```

- API-Einheiten finden Sie unter [DescribeNetworkInterfaceAttribute AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeNetworkInterfaces** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeNetworkInterfaces`.

CLI

AWS CLI

Um Ihre Netzwerkschnittstellen zu beschreiben

In diesem Beispiel werden alle Ihre Netzwerkschnittstellen beschrieben.

Befehl:

```
aws ec2 describe-network-interfaces
```

Ausgabe:

```
{
  "NetworkInterfaces": [
    {
      "Status": "in-use",
      "MacAddress": "02:2f:8f:b0:cf:75",
      "SourceDestCheck": true,
      "VpcId": "vpc-a01106c2",
      "Description": "my network interface",
      "Association": {
        "PublicIp": "203.0.113.12",
        "AssociationId": "eipassoc-0fbb766a",
        "PublicDnsName": "ec2-203-0-113-12.compute-1.amazonaws.com",
        "IpOwnerId": "123456789012"
      },
      "NetworkInterfaceId": "eni-e5aa89a3",
      "PrivateIpAddresses": [
        {
          "PrivateDnsName": "ip-10-0-1-17.ec2.internal",
          "Association": {
            "PublicIp": "203.0.113.12",
            "AssociationId": "eipassoc-0fbb766a",
            "PublicDnsName":
"ec2-203-0-113-12.compute-1.amazonaws.com",
            "IpOwnerId": "123456789012"
          },
          "Primary": true,
          "PrivateIpAddress": "10.0.1.17"
        }
      ],
      "RequesterManaged": false,
      "Ipv6Addresses": [],
      "PrivateDnsName": "ip-10-0-1-17.ec2.internal",
      "AvailabilityZone": "us-east-1d",
      "Attachment": {
        "Status": "attached",
        "DeviceIndex": 1,
        "AttachTime": "2013-11-30T23:36:42.000Z",
        "InstanceId": "i-1234567890abcdef0",
        "DeleteOnTermination": false,
        "AttachmentId": "eni-attach-66c4350a",
        "InstanceOwnerId": "123456789012"
      },
    },
  ],
}
```



```
    "Groups": [
      {
        "GroupName": "default",
        "GroupId": "sg-8637d3e3"
      }
    ],
    "SubnetId": "subnet-b61f49f0",
    "OwnerId": "123456789012",
    "TagSet": [],
    "PrivateIpAddress": "10.0.1.17"
  },
  {
    "Status": "in-use",
    "MacAddress": "02:58:f5:ef:4b:06",
    "SourceDestCheck": true,
    "VpcId": "vpc-a01106c2",
    "Description": "Primary network interface",
    "Association": {
      "PublicIp": "198.51.100.0",
      "IpOwnerId": "amazon"
    },
    "NetworkInterfaceId": "eni-f9ba99bf",
    "PrivateIpAddresses": [
      {
        "Association": {
          "PublicIp": "198.51.100.0",
          "IpOwnerId": "amazon"
        },
        "Primary": true,
        "PrivateIpAddress": "10.0.1.149"
      }
    ],
    "RequesterManaged": false,
    "Ipv6Addresses": [],
    "AvailabilityZone": "us-east-1d",
    "Attachment": {
      "Status": "attached",
      "DeviceIndex": 0,
      "AttachTime": "2013-11-30T23:35:33.000Z",
      "InstanceId": "i-0598c7d356eba48d7",
      "DeleteOnTermination": true,
      "AttachmentId": "eni-attach-1b9db777",
      "InstanceOwnerId": "123456789012"
    }
  },
}
```

```
    "Groups": [  
      {  
        "GroupName": "default",  
        "GroupId": "sg-8637d3e3"  
      }  
    ],  
    "SubnetId": "subnet-b61f49f0",  
    "OwnerId": "123456789012",  
    "TagSet": [],  
    "PrivateIpAddress": "10.0.1.149"  
  }  
]  
}
```

Dieses Beispiel beschreibt Netzwerkschnittstellen, die ein Tag mit dem Schlüssel `Purpose` und dem Wert `habenProd` haben.

Befehl:

```
aws ec2 describe-network-interfaces --filters Name=tag:Purpose,Values=Prod
```

Ausgabe:

```
{  
  "NetworkInterfaces": [  
    {  
      "Status": "available",  
      "MacAddress": "12:2c:bd:f9:bf:17",  
      "SourceDestCheck": true,  
      "VpcId": "vpc-8941ebec",  
      "Description": "ProdENI",  
      "NetworkInterfaceId": "eni-b9a5ac93",  
      "PrivateIpAddresses": [  
        {  
          "PrivateDnsName": "ip-10-0-1-55.ec2.internal",  
          "Primary": true,  
          "PrivateIpAddress": "10.0.1.55"  
        },  
        {  
          "PrivateDnsName": "ip-10-0-1-117.ec2.internal",  
          "Primary": false,  
          "PrivateIpAddress": "10.0.1.117"  
        }  
      ]  
    }  
  ]  
}
```

```

    }
  ],
  "RequesterManaged": false,
  "PrivateDnsName": "ip-10-0-1-55.ec2.internal",
  "AvailabilityZone": "us-east-1d",
  "Ipv6Addresses": [],
  "Groups": [
    {
      "GroupName": "MySG",
      "GroupId": "sg-905002f5"
    }
  ],
  "SubnetId": "subnet-31d6c219",
  "OwnerId": "123456789012",
  "TagSet": [
    {
      "Value": "Prod",
      "Key": "Purpose"
    }
  ],
  "PrivateIpAddress": "10.0.1.55"
}
]
}

```

- API-Einheiten finden Sie [DescribeNetworkInterfaces](#) unter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Netzwerkschnittstelle.

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

Ausgabe:

```

Association      :
Attachment       : Amazon.EC2.Model.NetworkInterfaceAttachment
AvailabilityZone : us-west-2c
Description      :
Groups           : {my-security-group}

```

```
MacAddress      : 0a:e9:a6:19:4c:7f
NetworkInterfaceId : eni-12345678
OwnerId        : 123456789012
PrivateDnsName  : ip-10-0-0-107.us-west-2.compute.internal
PrivateIpAddress : 10.0.0.107
PrivateIpAddresses : {ip-10-0-0-107.us-west-2.compute.internal}
RequesterId     :
RequesterManaged : False
SourceDestCheck : True
Status          : in-use
SubnetId        : subnet-1a2b3c4d
TagSet          : {}
VpcId           : vpc-12345678
```

Beispiel 2: Dieses Beispiel beschreibt alle Ihre Netzwerkschnittstellen.

```
Get-EC2NetworkInterface
```

- API-Einheiten finden Sie unter [DescribeNetworkInterfaces AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribePlacementGroups** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribePlacementGroups`.

CLI

AWS CLI

Um Ihre Platzierungsgruppen zu beschreiben

Dieser Beispielbefehl beschreibt alle Ihre Platzierungsgruppen.

Befehl:

```
aws ec2 describe-placement-groups
```

Ausgabe:

```
{
  "PlacementGroups": [
    {
      "GroupName": "my-cluster",
      "State": "available",
      "Strategy": "cluster"
    },
    ...
  ]
}
```

- API-Einheiten finden Sie [DescribePlacementGroups](#) in der AWS CLI Befehlsreferenz.

PowerShell**Tools für PowerShell**

Beispiel 1: Dieses Beispiel beschreibt die angegebene Platzierungsgruppe.

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

Ausgabe:

GroupName	State	Strategy
-----	-----	-----
my-placement-group	available	cluster

- API-Einheiten finden Sie unter [DescribePlacementGroups AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie `DescribePrefixLists` mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribePrefixLists`.

CLI

AWS CLI

Um Präfixlisten zu beschreiben

In diesem Beispiel werden alle verfügbaren Präfixlisten für die Region aufgeführt.

Befehl:

```
aws ec2 describe-prefix-lists
```

Ausgabe:

```
{
  "PrefixLists": [
    {
      "PrefixListName": "com.amazonaws.us-east-1.s3",
      "Cidrs": [
        "54.231.0.0/17"
      ],
      "PrefixListId": "pl-63a5400a"
    }
  ]
}
```

- API-Einheiten finden Sie [DescribePrefixLists](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das AWS-Services in einer Präfixliste verfügbare Format für die Region abgerufen

```
Get-EC2PrefixList
```

Ausgabe:

```
Cidrs                                PrefixListId PrefixListName
-----                                -
```

```
{52.94.5.0/24, 52.119.240.0/21, 52.94.24.0/23} pl-6fa54006 com.amazonaws.eu-
west-1.dynamodb
{52.218.0.0/17, 54.231.128.0/19} pl-6da54004 com.amazonaws.eu-
west-1.s3
```

- API-Einheiten finden Sie unter [DescribePrefixLists AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeRegions** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DescribeRegions`.

C++

SDK für C++

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#!/ Describe all Amazon Elastic Compute Cloud (Amazon EC2) Regions.
/*!
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeRegions(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::DescribeRegionsRequest request;
    Aws::EC2::Model::DescribeRegionsOutcome outcome =
    ec2Client.DescribeRegions(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
            std::setw(32) << "RegionName" <<
```

```
        std::setw(64) << "Endpoint" << std::endl;

    const auto &regions = outcome.GetResult().GetRegions();
    for (const auto &region: regions) {
        std::cout << std::left <<
            std::setw(32) << region.GetRegionName() <<
            std::setw(64) << region.GetEndpoint() << std::endl;
    }
} else {
    std::cerr << "Failed to describe regions:" <<
        outcome.GetError().GetMessage() << std::endl;
}

std::cout << std::endl;

return outcome.IsSuccess();
}
```

- API-Einheiten finden Sie [DescribeRegions](#) unter AWS SDK for C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie alle von Ihnen aktivierten Regionen

Im folgenden `describe-regions`-Beispiel werden alle Regionen beschrieben, die für Ihr Konto aktiviert sind.

```
aws ec2 describe-regions
```

Ausgabe:

```
{
  "Regions": [
    {
      "Endpoint": "ec2.eu-north-1.amazonaws.com",
      "RegionName": "eu-north-1",
      "OptInStatus": "opt-in-not-required"
    },
  ],
}
```



```
{
  "Endpoint": "ec2.ap-south-1.amazonaws.com",
  "RegionName": "ap-south-1",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.eu-west-3.amazonaws.com",
  "RegionName": "eu-west-3",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.eu-west-2.amazonaws.com",
  "RegionName": "eu-west-2",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.eu-west-1.amazonaws.com",
  "RegionName": "eu-west-1",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.ap-northeast-3.amazonaws.com",
  "RegionName": "ap-northeast-3",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.ap-northeast-2.amazonaws.com",
  "RegionName": "ap-northeast-2",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.ap-northeast-1.amazonaws.com",
  "RegionName": "ap-northeast-1",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.sa-east-1.amazonaws.com",
  "RegionName": "sa-east-1",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.ca-central-1.amazonaws.com",
  "RegionName": "ca-central-1",
  "OptInStatus": "opt-in-not-required"
}
```

```
    },
    {
      "Endpoint": "ec2.ap-southeast-1.amazonaws.com",
      "RegionName": "ap-southeast-1",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.ap-southeast-2.amazonaws.com",
      "RegionName": "ap-southeast-2",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.eu-central-1.amazonaws.com",
      "RegionName": "eu-central-1",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.us-east-1.amazonaws.com",
      "RegionName": "us-east-1",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.us-east-2.amazonaws.com",
      "RegionName": "us-east-2",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.us-west-1.amazonaws.com",
      "RegionName": "us-west-1",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.us-west-2.amazonaws.com",
      "RegionName": "us-west-2",
      "OptInStatus": "opt-in-not-required"
    }
  ]
}
```

Weitere Informationen finden Sie unter [Regionen und Zonen](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 2: So beschreiben Sie aktivierte Regionen mit einem Endpunkt, dessen Name eine bestimmte Zeichenfolge enthält

Das folgende `describe-regions`-Beispiel beschreibt alle Regionen, die Sie aktiviert haben und deren Endpunkt die Zeichenfolge „us“ enthält.

```
aws ec2 describe-regions \  
  --filters "Name=endpoint,Values=*us*"
```

Ausgabe:

```
{  
  "Regions": [  
    {  
      "Endpoint": "ec2.us-east-1.amazonaws.com",  
      "RegionName": "us-east-1"  
    },  
    {  
      "Endpoint": "ec2.us-east-2.amazonaws.com",  
      "RegionName": "us-east-2"  
    },  
    {  
      "Endpoint": "ec2.us-west-1.amazonaws.com",  
      "RegionName": "us-west-1"  
    },  
    {  
      "Endpoint": "ec2.us-west-2.amazonaws.com",  
      "RegionName": "us-west-2"  
    }  
  ]  
}
```

Weitere Informationen finden Sie unter [Regionen und Zonen](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 3: So beschreiben Sie alle Regionen

Das folgende `describe-regions`-Beispiel beschreibt alle verfügbaren Regionen, einschließlich Regionen, die deaktiviert sind.

```
aws ec2 describe-regions \  
  --all-regions
```

Ausgabe:

```
{
  "Regions": [
    {
      "Endpoint": "ec2.eu-north-1.amazonaws.com",
      "RegionName": "eu-north-1",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.ap-south-1.amazonaws.com",
      "RegionName": "ap-south-1",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.eu-west-3.amazonaws.com",
      "RegionName": "eu-west-3",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.eu-west-2.amazonaws.com",
      "RegionName": "eu-west-2",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.eu-west-1.amazonaws.com",
      "RegionName": "eu-west-1",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.ap-northeast-3.amazonaws.com",
      "RegionName": "ap-northeast-3",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.me-south-1.amazonaws.com",
      "RegionName": "me-south-1",
      "OptInStatus": "not-opted-in"
    },
    {
      "Endpoint": "ec2.ap-northeast-2.amazonaws.com",
      "RegionName": "ap-northeast-2",
      "OptInStatus": "opt-in-not-required"
    },
  ],
}
```

```
{
  "Endpoint": "ec2.ap-northeast-1.amazonaws.com",
  "RegionName": "ap-northeast-1",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.sa-east-1.amazonaws.com",
  "RegionName": "sa-east-1",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.ca-central-1.amazonaws.com",
  "RegionName": "ca-central-1",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.ap-east-1.amazonaws.com",
  "RegionName": "ap-east-1",
  "OptInStatus": "not-opted-in"
},
{
  "Endpoint": "ec2.ap-southeast-1.amazonaws.com",
  "RegionName": "ap-southeast-1",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.ap-southeast-2.amazonaws.com",
  "RegionName": "ap-southeast-2",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.eu-central-1.amazonaws.com",
  "RegionName": "eu-central-1",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.us-east-1.amazonaws.com",
  "RegionName": "us-east-1",
  "OptInStatus": "opt-in-not-required"
},
{
  "Endpoint": "ec2.us-east-2.amazonaws.com",
  "RegionName": "us-east-2",
  "OptInStatus": "opt-in-not-required"
}
```

```
    },
    {
      "Endpoint": "ec2.us-west-1.amazonaws.com",
      "RegionName": "us-west-1",
      "OptInStatus": "opt-in-not-required"
    },
    {
      "Endpoint": "ec2.us-west-2.amazonaws.com",
      "RegionName": "us-west-2",
      "OptInStatus": "opt-in-not-required"
    }
  ]
}
```

Weitere Informationen finden Sie unter [Regionen und Zonen](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 4: So listen Sie nur die Namen der Regionen auf

Im folgenden `describe-regions`-Beispiel wird der `--query`-Parameter verwendet, um die Ausgabe zu filtern und nur die Namen der Regionen als Text zurückzugeben.

```
aws ec2 describe-regions \
  --all-regions \
  --query "Regions[].{Name:RegionName}" \
  --output text
```

Ausgabe:

```
eu-north-1
ap-south-1
eu-west-3
eu-west-2
eu-west-1
ap-northeast-3
ap-northeast-2
me-south-1
ap-northeast-1
sa-east-1
ca-central-1
ap-east-1
ap-southeast-1
ap-southeast-2
```

```
eu-central-1
us-east-1
us-east-2
us-west-1
us-west-2
```

Weitere Informationen finden Sie unter [Regionen und Zonen](#) im EC2Amazon-Benutzerhandbuch.

- API-Einheiten finden Sie [DescribeRegions](#) in der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import { DescribeRegionsCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * List all available AWS regions.
 * @param {{ regionNames: string[], includeOptInRegions: boolean }} options
 */
export const main = async ({ regionNames, includeOptInRegions }) => {
  const client = new EC2Client({});
  const command = new DescribeRegionsCommand({
    // By default this command will not show regions that require you to opt-in.
    // When AllRegions is true, even the regions that require opt-in will be
    returned.
    AllRegions: includeOptInRegions,
    // You can omit the Filters property if you want to get all regions.
    Filters: regionNames?.length
      ? [
        {
          Name: "region-name",
          // You can specify multiple values for a filter.
          // You can also use '*' as a wildcard. This will return all
          // of the regions that start with `us-east-`.

```

```

        Values: regionNames,
      },
    ]
    : undefined,
  });

  try {
    const { Regions } = await client.send(command);
    const regionsList = Regions.map((reg) => ` • ${reg.RegionName}`);
    console.log("Found regions:");
    console.log(regionsList.join("\n"));
  } catch (caught) {
    if (caught instanceof Error && caught.name === "DryRunOperation") {
      console.log(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};

```

- API-Einheiten finden Sie [DescribeRegions](#) unter AWS SDK for JavaScript API-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Regionen beschrieben, die Ihnen zur Verfügung stehen.

```
Get-EC2Region
```

Ausgabe:

Endpoint	RegionName
-----	-----
ec2.eu-west-1.amazonaws.com	eu-west-1
ec2.ap-southeast-1.amazonaws.com	ap-southeast-1
ec2.ap-southeast-2.amazonaws.com	ap-southeast-2
ec2.eu-central-1.amazonaws.com	eu-central-1
ec2.ap-northeast-1.amazonaws.com	ap-northeast-1
ec2.us-east-1.amazonaws.com	us-east-1

ec2.sa-east-1.amazonaws.com	sa-east-1
ec2.us-west-1.amazonaws.com	us-west-1
ec2.us-west-2.amazonaws.com	us-west-2

- API-Einheiten finden Sie unter [DescribeRegions AWS Tools for PowerShell Cmdlet-Referenz](#).

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
require 'aws-sdk-ec2'

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
# list_regions_endpoints(Aws::EC2::Client.new(region: 'us-west-2'))
def list_regions_endpoints(ec2_client)
  result = ec2_client.describe_regions
  # Enable pretty printing.
  max_region_string_length = 16
  max_endpoint_string_length = 33
  # Print header.
  print 'Region'
  print ' ' * (max_region_string_length - 'Region'.length)
  print "  Endpoint\n"
  print '-' * max_region_string_length
  print ' '
  print '-' * max_endpoint_string_length
  print "\n"
  # Print Regions and their endpoints.
  result.regions.each do |region|
    print region.region_name
    print ' ' * (max_region_string_length - region.region_name.length)
    print ' '
  end
end
```

```
    print region.endpoint
    print "\n"
  end
end

# Displays a list of Amazon Elastic Compute Cloud (Amazon EC2)
# Availability Zones available to you depending on the AWS Region
# of the Amazon EC2 client.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_availability_zones(Aws::EC2::Client.new(region: 'us-west-2'))
def list_availability_zones(ec2_client)
  result = ec2_client.describe_availability_zones
  # Enable pretty printing.
  max_region_string_length = 16
  max_zone_string_length = 18
  max_state_string_length = 9
  # Print header.
  print 'Region'
  print ' ' * (max_region_string_length - 'Region'.length)
  print ' Zone'
  print ' ' * (max_zone_string_length - 'Zone'.length)
  print " State\n"
  print '-' * max_region_string_length
  print ' '
  print '-' * max_zone_string_length
  print ' '
  print '-' * max_state_string_length
  print "\n"
  # Print Regions, Availability Zones, and their states.
  result.availability_zones.each do |zone|
    print zone.region_name
    print ' ' * (max_region_string_length - zone.region_name.length)
    print ' '
    print zone.zone_name
    print ' ' * (max_zone_string_length - zone.zone_name.length)
    print ' '
    print zone.state
    # Print any messages for this Availability Zone.
    if zone.messages.count.positive?
      print "\n"
      puts ' Messages for this zone:'
      zone.messages.each do |message|
```

```
        print "    #{message.message}\n"
      end
    end
    print "\n"
  end
end

# Example usage:
def run_me
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-regions-availability-zones.rb REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-regions-availability-zones.rb us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts 'AWS Regions for Amazon EC2 that are available to you:'
  list_regions_endpoints(ec2_client)
  puts "\n\nAmazon EC2 Availability Zones that are available to you for AWS
Region '#{region}':"
  list_availability_zones(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- API Einheiten finden Sie [DescribeRegions](#) unter AWS SDK for Ruby API Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
async fn show_regions(client: &Client) -> Result<(), Error> {
    let rsp = client.describe_regions().send().await?;

    println!("Regions:");
    for region in rsp.regions() {
        println!("  {}", region.region_name().unwrap());
    }

    Ok(())
}
```

- API-Einheiten finden Sie [DescribeRegions](#) in der AWS SDK API Rust-Referenz.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
TRY.
    oo_result = lo_ec2->describeregions( ) .
    oo_result is returned for testing purposes. "
    DATA(lt_regions) = oo_result->get_regions( ).
    MESSAGE 'Retrieved information about Regions.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
```

```
DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-  
>av_err_msg }|.
MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- API-Einheiten finden Sie [DescribeRegions](#) unter AWS SDKSAPABAPI als Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeRouteTables** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DescribeRouteTables`.

CLI

AWS CLI

Um Ihre Routentabellen zu beschreiben

Im folgenden `describe-route-tables` Beispiel werden die Details zu Ihren Routentabellen abgerufen

```
aws ec2 describe-route-tables
```

Ausgabe:

```
{
  "RouteTables": [
    {
      "Associations": [
        {
          "Main": true,
          "RouteTableAssociationId": "rtbassoc-0df3f54e06EXAMPLE",
          "RouteTableId": "rtb-09ba434c1bEXAMPLE"
        }
      ],
      "PropagatingVgws": [],
      "RouteTableId": "rtb-09ba434c1bEXAMPLE",
```

```
"Routes": [
  {
    "DestinationCidrBlock": "10.0.0.0/16",
    "GatewayId": "local",
    "Origin": "CreateRouteTable",
    "State": "active"
  },
  {
    "DestinationCidrBlock": "0.0.0.0/0",
    "NatGatewayId": "nat-06c018cbd8EXAMPLE",
    "Origin": "CreateRoute",
    "State": "blackhole"
  }
],
"Tags": [],
"VpcId": "vpc-0065acced4EXAMPLE",
"OwnerId": "111122223333"
},
{
  "Associations": [
    {
      "Main": true,
      "RouteTableAssociationId": "rtbassoc-9EXAMPLE",
      "RouteTableId": "rtb-a1eec7de"
    }
  ],
  "PropagatingVgws": [],
  "RouteTableId": "rtb-a1eec7de",
  "Routes": [
    {
      "DestinationCidrBlock": "172.31.0.0/16",
      "GatewayId": "local",
      "Origin": "CreateRouteTable",
      "State": "active"
    },
    {
      "DestinationCidrBlock": "0.0.0.0/0",
      "GatewayId": "igw-fEXAMPLE",
      "Origin": "CreateRoute",
      "State": "active"
    }
  ],
  "Tags": [],
  "VpcId": "vpc-3EXAMPLE",
```

```

    "OwnerId": "111122223333"
  },
  {
    "Associations": [
      {
        "Main": false,
        "RouteTableAssociationId": "rtbassoc-0b100c28b2EXAMPLE",
        "RouteTableId": "rtb-07a98f76e5EXAMPLE",
        "SubnetId": "subnet-0d3d002af8EXAMPLE"
      }
    ],
    "PropagatingVgws": [],
    "RouteTableId": "rtb-07a98f76e5EXAMPLE",
    "Routes": [
      {
        "DestinationCidrBlock": "10.0.0.0/16",
        "GatewayId": "local",
        "Origin": "CreateRouteTable",
        "State": "active"
      },
      {
        "DestinationCidrBlock": "0.0.0.0/0",
        "GatewayId": "igw-06cf664d80EXAMPLE",
        "Origin": "CreateRoute",
        "State": "active"
      }
    ],
    "Tags": [],
    "VpcId": "vpc-0065acced4EXAMPLE",
    "OwnerId": "111122223333"
  }
]
}


```

Weitere Informationen finden Sie unter [Arbeiten mit Routentabellen](#) im AWS VPCBenutzerhandbuch.

- APIEinzelheiten finden Sie [DescribeRouteTables](#) unter AWS CLI Befehlsreferenz.

PHP

SDK für PHP

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * @param array $routeTableIds
 * @param array $filters
 * @return array
 */
public function describeRouteTables(array $routeTableIds = [], array $filters
= []): array
{
    $parameters = [];
    if($routeTableIds){
        $parameters['RouteTableIds'] = $routeTableIds;
    }
    if($filters){
        $parameters['Filters'] = $filters;
    }
    try {
        $paginator = $this->ec2Client->getPaginator("DescribeRouteTables",
$parameters);
        $contents = [];
        foreach ($paginator as $result) {
            foreach ($result['RouteTables'] as $object) {
                $contents[] = $object['RouteTableId'];
            }
        }
    }catch (Ec2Exception $caught){
        echo "There was a problem paginating the results of
DescribeRouteTables: {$caught->getAwsErrorMessage()}\n";
        throw $caught;
    }
    return $contents;
}
```


- API-Einheiten finden Sie [DescribeRouteTables](#) unter AWS SDK for PHP API-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt alle Ihre Routentabellen.

```
Get-EC2RouteTable
```

Ausgabe:

```
DestinationCidrBlock    : 10.0.0.0/16
DestinationPrefixListId :
GatewayId               : local
InstanceId              :
InstanceOwnerId         :
NetworkInterfaceId     :
Origin                  : CreateRouteTable
State                   : active
VpcPeeringConnectionId :

DestinationCidrBlock    : 0.0.0.0/0
DestinationPrefixListId :
GatewayId               : igw-1a2b3c4d
InstanceId              :
InstanceOwnerId         :
NetworkInterfaceId     :
Origin                  : CreateRoute
State                   : active
VpcPeeringConnectionId :
```

Beispiel 2: In diesem Beispiel werden Details für die angegebene Routentabelle zurückgegeben.

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

Beispiel 3: Dieses Beispiel beschreibt die Routentabellen für die angegebene VPC.

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

Ausgabe:

```
Associations      : {rtbassoc-12345678}
PropagatingVgws  : {}
Routes           : {, }
RouteTableId     : rtb-1a2b3c4d
Tags             : {}
VpcId            : vpc-1a2b3c4d
```

- API-Einheiten finden Sie unter [DescribeRouteTables AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeScheduledInstanceAvailability** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung von `DescribeScheduledInstanceAvailability`.

CLI

AWS CLI

Um einen verfügbaren Zeitplan zu beschreiben

Dieses Beispiel beschreibt einen Zeitplan, der jede Woche am Sonntag beginnt und am angegebenen Datum beginnt.

Befehl:

```
aws ec2 describe-scheduled-instance-availability --
recurrence Frequency=Weekly,Interval=1,OccurrenceDays=[1] --first-slot-start-
time-range EarliestTime=2016-01-31T00:00:00Z,LatestTime=2016-01-31T04:00:00Z
```

Ausgabe:

```
{
  "ScheduledInstanceAvailabilitySet": [
    {
      "AvailabilityZone": "us-west-2b",
      "TotalScheduledInstanceHours": 1219,
      "PurchaseToken": "eyJ2IjoiMSIsInMiOjEsImMiOi...",
      "MinTermDurationInDays": 366,
      "AvailableInstanceCount": 20,
      "Recurrence": {
        "OccurrenceDaySet": [
          1
        ],
        "Interval": 1,
        "Frequency": "Weekly",
        "OccurrenceRelativeToEnd": false
      },
      "Platform": "Linux/UNIX",
      "FirstSlotStartTime": "2016-01-31T00:00:00Z",
      "MaxTermDurationInDays": 366,
      "SlotDurationInHours": 23,
      "NetworkPlatform": "EC2-VPC",
      "InstanceType": "c4.large",
      "HourlyPrice": "0.095"
    },
    ...
  ]
}
```

Um die Ergebnisse einzugrenzen, können Sie Filter hinzufügen, die das Betriebssystem, das Netzwerk und den Instanztyp angeben.

Befehl:

```
--filters Name=Plattform, Werte=Linux/Name=Netzwerkplattform, Werte= - Name=Instanztyp,
Werte=C4.large UNIX EC2 VPC
```

- API-Einheiten finden Sie in der [DescribeScheduledInstanceAvailability AWS CLI-Befehlsreferenz](#).

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt einen Zeitplan, der jede Woche am Sonntag beginnt und am angegebenen Datum beginnt.

```
Get-EC2ScheduledInstanceAvailability -Recurrence_Frequency
  Weekly -Recurrence_Interval 1 -Recurrence_OccurrenceDay 1 -
FirstSlotStartTimeRange_EarliestTime 2016-01-31T00:00:00Z -
FirstSlotStartTimeRange_LatestTime 2016-01-31T04:00:00Z
```

Ausgabe:

```
AvailabilityZone           : us-west-2b
AvailableInstanceCount    : 20
FirstSlotStartTime        : 1/31/2016 8:00:00 AM
HourlyPrice                : 0.095
InstanceType              : c4.large
MaxTermDurationInDays     : 366
MinTermDurationInDays     : 366
NetworkPlatform           : EC2-VPC
Platform                  : Linux/UNIX
PurchaseToken             : eyJ2IjoiMSIsInMiOjEsImMiOi...
Recurrence                 : Amazon.EC2.Model.ScheduledInstanceRecurrence
SlotDurationInHours       : 23
TotalScheduledInstanceHours : 1219
...
```

Beispiel 2: Um die Ergebnisse einzugrenzen, können Sie Filter für Kriterien wie Betriebssystem, Netzwerk und Instanztyp hinzufügen.

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- API-Einheiten finden Sie unter [DescribeScheduledInstanceAvailability AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter. [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeScheduledInstances** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeScheduledInstances`.

CLI

AWS CLI

Um Ihre geplanten Instances zu beschreiben

Dieses Beispiel beschreibt die angegebene Scheduled Instance.

Befehl:

```
aws ec2 describe-scheduled-instances --scheduled-instance-ids sci-1234-1234-1234-1234-123456789012
```

Ausgabe:

```
{
  "ScheduledInstanceSet": [
    {
      "AvailabilityZone": "us-west-2b",
      "ScheduledInstanceId": "sci-1234-1234-1234-1234-123456789012",
      "HourlyPrice": "0.095",
      "CreateDate": "2016-01-25T21:43:38.612Z",
      "Recurrence": {
        "OccurrenceDaySet": [
          1
        ],
        "Interval": 1,
        "Frequency": "Weekly",
        "OccurrenceRelativeToEnd": false,
        "OccurrenceUnit": ""
      },
      "Platform": "Linux/UNIX",
      "TermEndDate": "2017-01-31T09:00:00Z",
      "InstanceCount": 1,
      "SlotDurationInHours": 32,
    }
  ]
}
```

```

        "TermStartDate": "2016-01-31T09:00:00Z",
        "NetworkPlatform": "EC2-VPC",
        "TotalScheduledInstanceHours": 1696,
        "NextSlotStartTime": "2016-01-31T09:00:00Z",
        "InstanceType": "c4.large"
    }
]
}

```

Dieses Beispiel beschreibt alle Ihre geplanten Instances.

Befehl:

```
aws ec2 describe-scheduled-instances
```

- API-Einheiten finden Sie [DescribeScheduledInstances](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene geplante Instanz.

```
Get-EC2ScheduledInstance -ScheduledInstanceId
sci-1234-1234-1234-1234-123456789012
```

Ausgabe:

```

AvailabilityZone      : us-west-2b
CreateDate            : 1/25/2016 1:43:38 PM
HourlyPrice          : 0.095
InstanceCount        : 1
InstanceType         : c4.large
NetworkPlatform      : EC2-VPC
NextSlotStartTime    : 1/31/2016 1:00:00 AM
Platform             : Linux/UNIX
PreviousSlotEndTime  :
Recurrence           : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId  : sci-1234-1234-1234-1234-123456789012
SlotDurationInHours  : 32
TermEndDate          : 1/31/2017 1:00:00 AM
TermStartDate        : 1/31/2016 1:00:00 AM

```

```
TotalScheduledInstanceHours : 1696
```

Beispiel 2: Dieses Beispiel beschreibt all Ihre geplanten Instances.

```
Get-EC2ScheduledInstance
```

- API-Einheiten finden Sie unter [DescribeScheduledInstances AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeSecurityGroups** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DescribeSecurityGroups`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Retrieve information for one or all Amazon EC2 security group.
/// </summary>
/// <param name="groupId">The optional Id of a specific Amazon EC2 security
group.</param>
/// <returns>A list of security group information.</returns>
public async Task<List<SecurityGroup>> DescribeSecurityGroups(string groupId)
{
```

```
try
{
    var securityGroups = new List<SecurityGroup>();
    var request = new DescribeSecurityGroupsRequest();

    if (!string.IsNullOrEmpty(groupId))
    {
        var groupIds = new List<string> { groupId };
        request.GroupIds = groupIds;
    }

    var paginatorForSecurityGroups =
        _amazonEC2.Paginators.DescribeSecurityGroups(request);

    await foreach (var securityGroup in
paginatorForSecurityGroups.SecurityGroups)
    {
        securityGroups.Add(securityGroup);
    }

    return securityGroups;
}
catch (AmazonEC2Exception ec2Exception)
{
    if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
    {
        _logger.LogError(
            $"A security group {groupId} does not exist.");
    }

    throw;
}
catch (Exception ex)
{
    _logger.LogError(
        $"An error occurred while listing security groups.
{ex.Message}");
    throw;
}
}

/// <summary>
/// Display the information returned by the call to
```



```
/// DescribeSecurityGroupsAsync.
/// </summary>
/// <param name="securityGroup">A list of security group information.</param>
public void DisplaySecurityGroupInfoAsync(SecurityGroup securityGroup)
{
    Console.WriteLine($"{securityGroup.GroupName}");
    Console.WriteLine("Ingress permissions:");
    securityGroup.IpPermissions.ForEach(permission =>
    {
        Console.WriteLine($"  \tFromPort: {permission.FromPort}");
        Console.WriteLine($"  \tIpProtocol: {permission.IpProtocol}");

        Console.WriteLine($"  \tIpv4Ranges: ");
        permission.Ipv4Ranges.ForEach(range =>
        { Console.WriteLine($"  \t{range.CidrIp} "); });

        Console.WriteLine($"  \n\tIpv6Ranges:");
        permission.Ipv6Ranges.ForEach(range =>
        { Console.WriteLine($"  \t{range.CidrIpv6} "); });

        Console.WriteLine($"  \n\tPrefixListIds: ");
        permission.PrefixListIds.ForEach(id => Console.WriteLine($"  \t{id.Id} "));

        Console.WriteLine($"  \n\tTo Port: {permission.ToPort}");
    });
    Console.WriteLine("Egress permissions:");
    securityGroup.IpPermissionsEgress.ForEach(permission =>
    {
        Console.WriteLine($"  \tFromPort: {permission.FromPort}");
        Console.WriteLine($"  \tIpProtocol: {permission.IpProtocol}");

        Console.WriteLine($"  \tIpv4Ranges: ");
        permission.Ipv4Ranges.ForEach(range =>
        { Console.WriteLine($"  \t{range.CidrIp} "); });

        Console.WriteLine($"  \n\tIpv6Ranges:");
        permission.Ipv6Ranges.ForEach(range =>
        { Console.WriteLine($"  \t{range.CidrIpv6} "); });

        Console.WriteLine($"  \n\tPrefixListIds: ");
        permission.PrefixListIds.ForEach(id => Console.WriteLine($"  \t{id.Id} "));

        Console.WriteLine($"  \n\tTo Port: {permission.ToPort}");
    });
}
```

```
}

```

- API-Einheiten finden Sie [DescribeSecurityGroups](#) unter AWS SDK for .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#####
# function ec2_describe_security_groups
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
# security groups.
#
# Parameters:
#   -g security_group_id - The ID of the security group to describe
#   (optional).
#
# And:
#   0 - If successful.
#   1 - If it fails.
#####
function ec2_describe_security_groups() {
    local security_group_id response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_describe_security_groups"
        echo "Describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
security groups."
        echo "  -g security_group_id - The ID of the security group to describe
(optional)."
        echo ""
    }
}

```

```
}

# Retrieve the calling parameters.
while getopts "g:h" option; do
  case "${option}" in
    g) security_group_id="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

local query="SecurityGroups[*].[GroupName, GroupId, VpcId, IpPermissions[*].
[IpProtocol, FromPort, ToPort, IpRanges[*].CidrIp]]"

if [[ -n "$security_group_id" ]]; then
  response=$(aws ec2 describe-security-groups --group-ids "$security_group_id"
--query "${query}" --output text)
else
  response=$(aws ec2 describe-security-groups --query "${query}" --output text)
fi

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports describe-security-groups operation failed.
$response"
  return 1
fi

echo "$response"

return 0
}
```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```
#####  
# function errecho  
#  
# This function outputs everything sent to it to STDERR (standard error output).  
#####  
function errecho() {  
    printf "%s\n" "$*" 1>&2  
}  
  
#####  
# function aws_cli_error_log()  
#  
# This function is used to log the error messages from the AWS CLI.  
#  
# The function expects the following argument:  
#     $1 - The error code returned by the AWS CLI.  
#  
# Returns:  
#     0: - Success.  
#  
#####  
function aws_cli_error_log() {  
    local err_code=$1  
    errecho "Error code : $err_code"  
    if [ "$err_code" == 1 ]; then  
        errecho " One or more S3 transfers failed."  
    elif [ "$err_code" == 2 ]; then  
        errecho " Command line failed to parse."  
    elif [ "$err_code" == 130 ]; then  
        errecho " Process received SIGINT."  
    elif [ "$err_code" == 252 ]; then  
        errecho " Command syntax invalid."  
    elif [ "$err_code" == 253 ]; then  
        errecho " The system environment or configuration was invalid."  
    elif [ "$err_code" == 254 ]; then  
        errecho " The service returned an error."  
    elif [ "$err_code" == 255 ]; then  
        errecho " 255 is a catch-all error."  
    fi  
  
    return 0  
}
```

- API-Einheiten finden Sie [DescribeSecurityGroups](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#!/ Describe all Amazon Elastic Compute Cloud (Amazon EC2) security groups, or a
specific group.
/*!
  \param groupID: A group ID, ignored if empty.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeSecurityGroups(const Aws::String &groupID,
                                         const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeSecurityGroupsRequest request;

    if (!groupID.empty()) {
        request.AddGroupIds(groupID);
    }

    Aws::String nextToken;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::EC2::Model::DescribeSecurityGroupsOutcome outcome =
ec2Client.DescribeSecurityGroups(request);
        if (outcome.IsSuccess()) {
            std::cout << std::left <<
                std::setw(32) << "Name" <<
```

```

        std::setw(30) << "GroupId" <<
        std::setw(30) << "VpcId" <<
        std::setw(64) << "Description" << std::endl;

    const std::vector<Aws::EC2::Model::SecurityGroup> &securityGroups =
        outcome.GetResult().GetSecurityGroups();

    for (const auto &securityGroup: securityGroups) {
        std::cout << std::left <<
            std::setw(32) << securityGroup.GetGroupName() <<
            std::setw(30) << securityGroup.GetGroupId() <<
            std::setw(30) << securityGroup.GetVpcId() <<
            std::setw(64) << securityGroup.GetDescription() <<
            std::endl;
    }
} else {
    std::cerr << "Failed to describe security groups:" <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

    nextToken = outcome.GetResult().GetNextToken();
} while (!nextToken.empty());

return true;
}

```

- API-Einheiten finden Sie [DescribeSecurityGroups](#) unter AWS SDK for C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie eine Sicherheitsgruppe

Das folgende `describe-security-groups`-Beispiel beschreibt die angegebene Sicherheitsgruppe.

```

aws ec2 describe-security-groups \
    --group-ids sg-903004f8

```

Ausgabe:

```
{
  "SecurityGroups": [
    {
      "IpPermissionsEgress": [
        {
          "IpProtocol": "-1",
          "IpRanges": [
            {
              "CidrIp": "0.0.0.0/0"
            }
          ],
          "UserIdGroupPairs": [],
          "PrefixListIds": []
        }
      ],
      "Description": "My security group",
      "Tags": [
        {
          "Value": "SG1",
          "Key": "Name"
        }
      ],
      "IpPermissions": [
        {
          "IpProtocol": "-1",
          "IpRanges": [],
          "UserIdGroupPairs": [
            {
              "UserId": "123456789012",
              "GroupId": "sg-903004f8"
            }
          ],
          "PrefixListIds": []
        }
      ],
      {
        "PrefixListIds": [],
        "FromPort": 22,
        "IpRanges": [
          {
            "Description": "Access from NY office",
            "CidrIp": "203.0.113.0/24"
          }
        ]
      }
    }
  ]
}
```

```

        ],
        "ToPort": 22,
        "IpProtocol": "tcp",
        "UserIdGroupPairs": []
      }
    ],
    "GroupName": "MySecurityGroup",
    "VpcId": "vpc-1a2b3c4d",
    "OwnerId": "123456789012",
    "GroupId": "sg-903004f8",
  }
]
}

```

Beispiel 2: So beschreiben Sie Sicherheitsgruppen, die bestimmte Regeln haben

Im folgenden `describe-security-groups` Beispiel werden Filter verwendet, um die Ergebnisse auf Sicherheitsgruppen zu beschränken, die eine Regel haben, die SSH Datenverkehr zulässt (Port 22), und eine Regel, die Datenverkehr von allen Adressen zulässt (`0.0.0.0/0`). Im Beispiel wird der `--query`-Parameter verwendet, um nur die Namen der Sicherheitsgruppen anzuzeigen. Sicherheitsgruppen müssen mit allen Filtern übereinstimmen, die in den Ergebnissen zurückgegeben werden. Eine einzige Regel muss jedoch nicht mit allen Filtern übereinstimmen. Die Ausgabe gibt beispielsweise eine Sicherheitsgruppe mit einer Regel zurück, die SSH Datenverkehr von einer bestimmten IP-Adresse zulässt, und einer anderen Regel, die HTTP Datenverkehr von allen Adressen zulässt.

```

aws ec2 describe-security-groups \
  --filters Name=ip-permission.from-port,Values=22 Name=ip-permission.to-
port,Values=22 Name=ip-permission.cidr,Values='0.0.0.0/0' \
  --query "SecurityGroups[*].[GroupName]" \
  --output text

```

Ausgabe:

```

default
my-security-group
web-servers
launch-wizard-1

```

Beispiel 3: So beschreiben Sie Sicherheitsgruppen anhand von Tags

Im folgenden `describe-security-groups`-Beispiel werden Filter verwendet, um die Ergebnisse auf Sicherheitsgruppen zu beschränken, die `test` im Namen der Sicherheitsgruppe enthalten und die das Tag `Test=To-delete` haben. Das Beispiel verwendet den `--query` Parameter, um nur die Namen und die IDs der Sicherheitsgruppen anzuzeigen.

```
aws ec2 describe-security-groups \  
  --filters Name=group-name,Values=*test* Name=tag:Test,Values=To-delete \  
  --query "SecurityGroups[*].{Name:GroupName,ID:GroupId}"
```

Ausgabe:

```
[  
  {  
    "Name": "testfornewinstance",  
    "ID": "sg-33bb22aa"  
  },  
  {  
    "Name": "newgrouptest",  
    "ID": "sg-1a2b3c4d"  
  }  
]
```

Weitere Beispiele für die Verwendung von Tag-Filtern finden Sie unter [Arbeiten mit Tags](#) im EC2Amazon-Benutzerhandbuch.

- API-Einheiten finden Sie [DescribeSecurityGroups](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Asynchronously describes the security groups for the specified group ID.
 *
 * @param groupName the name of the security group to describe
 * @return a {@link CompletableFuture} that represents the asynchronous
operation
 *         of describing the security groups. The future will complete with a
 *         {@link DescribeSecurityGroupsResponse} object that contains the
 *         security group information.
 */
public CompletableFuture<String> describeSecurityGroupArnByNameAsync(String
groupName) {
    DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
        .groupNames(groupName)
        .build();

    DescribeSecurityGroupsPublisher paginator =
getAsyncClient().describeSecurityGroupsPaginator(request);
    AtomicReference<String> groupIdRef = new AtomicReference<>();
    return paginator.subscribe(response -> {
        response.securityGroups().stream()
            .filter(securityGroup ->
securityGroup.groupName().equals(groupName))
            .findFirst()
            .ifPresent(securityGroup ->
groupIdRef.set(securityGroup.groupId()));
    }).thenApply(v -> {
        String groupId = groupIdRef.get();
        if (groupId == null) {
            throw new RuntimeException("No security group found with the
name: " + groupName);
        }
        return groupId;
    }).exceptionally(ex -> {
        logger.info("Failed to describe security group: " + ex.getMessage());
        throw new RuntimeException("Failed to describe security group", ex);
    });
}
```

- APIEinzelheiten finden Sie [DescribeSecurityGroups](#) unter AWS SDK for Java 2.x APIReferenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import { DescribeSecurityGroupsCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Describes the specified security groups or all of your security groups.
 * @param {{ groupIds: string[] }} options
 */
export const main = async ({ groupIds = [] }) => {
  const client = new EC2Client({});
  const command = new DescribeSecurityGroupsCommand({
    GroupIds: groupIds,
  });

  try {
    const { SecurityGroups } = await client.send(command);
    const sgList = SecurityGroups.map(
      (sg) => `• ${sg.GroupName} (${sg.GroupId}): ${sg.Description}`,
    ).join("\n");
    if (sgList.length) {
      console.log(`Security groups:\n${sgList}`);
    } else {
      console.log("No security groups found.");
    }
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
      console.warn(`${caught.message}. Please provide a valid GroupId.`);
    } else if (
      caught instanceof Error &&
      caught.name === "InvalidGroup.NotFound"
    ) {
      console.warn(caught.message);
    } else {
      throw caught;
    }
  }
}
```

```
    }  
  }  
};
```

- APIEinzelheiten finden Sie [DescribeSecurityGroups](#) unter AWS SDK for JavaScript APIReferenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
suspend fun describeEC2SecurityGroups(groupId: String) {  
    val request =  
        DescribeSecurityGroupsRequest {  
            groupIds = listOf(groupId)  
        }  
  
    Ec2Client { region = "us-west-2" }.use { ec2 ->  
  
        val response = ec2.describeSecurityGroups(request)  
        response.securityGroups?.forEach { group ->  
            println("Found Security Group with id ${group.groupId}, vpc id  
${group.vpcId} and description ${group.description}")  
        }  
    }  
}
```

- APIEinzelheiten finden Sie [DescribeSecurityGroups](#) in der AWS SDKAPIKotlin-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Sicherheitsgruppe für VPC a. Wenn Sie mit Sicherheitsgruppen arbeiten, die zu einer gehören, müssen VPC Sie die Sicherheitsgruppen-ID (- GroupId Parameter) und nicht den Namen (- GroupName Parameter) verwenden, um auf die Gruppe zu verweisen.

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

Ausgabe:

```
Description      : default VPC security group
GroupId          : sg-12345678
GroupName       : default
IpPermissions    : {Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {Amazon.EC2.Model.IpPermission}
OwnerId         : 123456789012
Tags            : {}
VpcId           : vpc-12345678
```

Beispiel 2: Dieses Beispiel beschreibt die angegebene Sicherheitsgruppe für EC2 -Classic. Wenn Sie mit Sicherheitsgruppen für EC2 -Classic arbeiten, können Sie entweder den Gruppennamen (- GroupName Parameter) oder die Gruppen-ID (- GroupId Parameter) verwenden, um auf die Sicherheitsgruppe zu verweisen.

```
Get-EC2SecurityGroup -GroupName my-security-group
```

Ausgabe:

```
Description      : my security group
GroupId          : sg-45678901
GroupName       : my-security-group
IpPermissions    : {Amazon.EC2.Model.IpPermission,
  Amazon.EC2.Model.IpPermission}
IpPermissionsEgress : {}
OwnerId         : 123456789012
Tags            : {}
VpcId           :
```

Beispiel 3: In diesem Beispiel werden alle Sicherheitsgruppen für die Datei vpc-0fc1ff23456b789eb abgerufen

```
Get-EC2SecurityGroup -Filter @{Name="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- API-Einheiten [DescribeSecurityGroups](#) finden AWS Tools for PowerShell Sie unter Cmdlet-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
    actions."""

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
    None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
        security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
        high-level identifier
                               that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
```

```
Creates a SecurityGroupWrapper instance with a default EC2 client.

:return: An instance of SecurityGroupWrapper initialized with the default
EC2 client.
"""
ec2_client = boto3.client("ec2")
return cls(ec2_client)

def describe(self, security_group_id: Optional[str] = None) -> bool:
    """
    Displays information about the specified security group or all security
    groups if no ID is provided.

    :param security_group_id: The ID of the security group to describe.
        If None, an open search is performed to
    describe all security groups.
    :returns: True if the description is successful.
    :raises ClientError: If there is an error describing the security
    group(s), such as an invalid security group ID.
    """
    try:
        paginator = self.ec2_client.get_paginator("describe_security_groups")

        if security_group_id is None:
            # If no ID is provided, return all security groups.
            page_iterator = paginator.paginate()
        else:
            page_iterator = paginator.paginate(GroupIds=[security_group_id])

        for page in page_iterator:
            for security_group in page["SecurityGroups"]:
                print(f"Security group: {security_group['GroupName']}")
                print(f"\tID: {security_group['GroupId']}")
                print(f"\tVPC: {security_group['VpcId']}")
                if security_group["IpPermissions"]:
                    print("Inbound permissions:")
                    pp(security_group["IpPermissions"])

        return True
    except ClientError as err:
        logger.error("Failed to describe security group(s).")
        if err.response["Error"]["Code"] == "InvalidGroup.NotFound":
            logger.error(
```

```

        f"Security group {security_group_id} does not exist "
        f"because the specified security group ID was not found."
    )
    raise

```

- API-Einheiten finden Sie unter [DescribeSecurityGroups](#) Python (Boto3) API - Referenz.AWS SDK

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

async fn show_security_groups(client: &aws_sdk_ec2::Client, group_ids:
Vec<String>) {
    let response = client
        .describe_security_groups()
        .set_group_ids(Some(group_ids))
        .send()
        .await;

    match response {
        Ok(output) => {
            for group in output.security_groups() {
                println!(
                    "Found Security Group {} ({}), vpc id {} and description {}",
                    group.group_name().unwrap_or("unknown"),
                    group.group_id().unwrap_or("id-unknown"),
                    group.vpc_id().unwrap_or("vpcid-unknown"),
                    group.description().unwrap_or("(none)")
                );
            }
        }
        Err(err) => {

```



```

        let err = err.into_service_error();
        let meta = err.meta();
        let message = meta.message().unwrap_or("unknown");
        let code = meta.code().unwrap_or("unknown");
        eprintln!("Error listing EC2 Security Groups: ({code}) {message}");
    }
}
}

```

- API-Einheiten finden Sie [DescribeSecurityGroups](#) in der AWS SDK API Rust-Referenz.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

TRY.
    DATA lt_group_ids TYPE /aws1/
cl_ec2groupidstrlist_w=>tt_groupidstringlist.
    APPEND NEW /aws1/cl_ec2groupidstrlist_w( iv_value = iv_group_id ) TO
lt_group_ids.
    oo_result = lo_ec2->describesecuritygroups( it_groupids = lt_group_ids ).
    " oo_result is returned for testing purposes. "
    DATA(lt_security_groups) = oo_result->get_securitygroups( ).
    MESSAGE 'Retrieved information about security groups.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- API-Einheiten finden Sie [DescribeSecurityGroups](#) unter AWS SDK SAP ABAP API als Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeSnapshotAttribute** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeSnapshotAttribute`.

CLI

AWS CLI

Um die Snapshot-Attribute für einen Snapshot zu beschreiben

Das folgende `describe-snapshot-attribute` Beispiel listet die Konten auf, mit denen ein Snapshot gemeinsam genutzt wird.

```
aws ec2 describe-snapshot-attribute \  
  --snapshot-id snap-01234567890abcdef \  
  --attribute createVolumePermission
```

Ausgabe:

```
{  
  "SnapshotId": "snap-01234567890abcdef",  
  "CreateVolumePermissions": [  
    {  
      "UserId": "123456789012"  
    }  
  ]  
}
```

Weitere Informationen finden Sie unter [Einen EBS Amazon-Snapshot teilen](#) im Amazon Elastic Compute Cloud-Benutzerhandbuch.

- API-Einheiten finden Sie [DescribeSnapshotAttribute](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das angegebene Attribut des angegebenen Snapshots.

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

Ausgabe:

```
CreateVolumePermissions    ProductCodes    SnapshotId
-----
{}                          {}               snap-12345678
```

Beispiel 2: Dieses Beispiel beschreibt das angegebene Attribut des angegebenen Snapshots.

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
CreateVolumePermission).CreateVolumePermissions
```

Ausgabe:

```
Group    UserId
-----
all
```

- API-Einheiten finden Sie unter [DescribeSnapshotAttribute AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeSnapshots** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DescribeSnapshots`.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie einen Snapshot

Das folgende `describe-snapshots`-Beispiel beschreibt den angegebenen Snapshot.

```
aws ec2 describe-snapshots \
  --snapshot-ids snap-1234567890abcdef0
```

Ausgabe:

```
{
  "Snapshots": [
    {
      "Description": "This is my snapshot",
      "Encrypted": false,
      "VolumeId": "vol-049df61146c4d7901",
      "State": "completed",
      "VolumeSize": 8,
      "StartTime": "2019-02-28T21:28:32.000Z",
      "Progress": "100%",
      "OwnerId": "012345678910",
      "SnapshotId": "snap-01234567890abcdef",
      "Tags": [
        {
          "Key": "Stack",
          "Value": "test"
        }
      ]
    }
  ]
}
```

Weitere Informationen finden Sie unter [EBSAmazon-Snapshots](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 2: So beschreiben Sie Snapshots basierend auf Filtern

Im folgenden `describe-snapshots` Beispiel werden Filter verwendet, um die Ergebnisse auf Snapshots zu beschränken, die Ihrem AWS Konto gehören und sich im `pending` Bundesstaat befinden. Im Beispiel wird der `--query` Parameter verwendet, um nur den Snapshot IDs und die Uhrzeit anzuzeigen, zu der der Snapshot gestartet wurde.

```
aws ec2 describe-snapshots \
  --owner-ids self \
  --filters Name=status,Values=pending \
  --query "Snapshots[*].{ID:SnapshotId,Time:StartTime}"
```

Ausgabe:

```
[
```

```
{
  "ID": "snap-1234567890abcdef0",
  "Time": "2019-08-04T12:48:18.000Z"
},
{
  "ID": "snap-066877671789bd71b",
  "Time": "2019-08-04T02:45:16.000Z"
},
...
]
```

Im folgenden `describe-snapshots`-Beispiel werden Filter verwendet, um die Ergebnisse auf Snapshots zu beschränken, die aus dem angegebenen Volume erstellt wurden. Das Beispiel verwendet den `--query` Parameter, um nur den Snapshot anzuzeigenIDs.

```
aws ec2 describe-snapshots \
  --filters Name=volume-id,Values=049df61146c4d7901 \
  --query "Snapshots[*].[SnapshotId]" \
  --output text
```

Ausgabe:

```
snap-1234567890abcdef0
snap-08637175a712c3fb9
...
```

Weitere Beispiele für die Verwendung von Filtern finden Sie unter [Ressourcen auflisten und filtern](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 3: So beschreiben Sie Snapshots auf der Grundlage von Tags

Im folgenden `describe-snapshots`-Beispiel werden Tag-Filter verwendet, um die Ergebnisse auf Snapshots zu beschränken, die das Tag `Stack=Prod` enthalten.

```
aws ec2 describe-snapshots \
  --filters Name=tag:Stack,Values=prod
```

Ein Beispiel für die Ausgabe von `describe-snapshots` finden Sie in Beispiel 1.

Weitere Beispiele für die Verwendung von Tag-Filtern finden Sie unter [Arbeiten mit Tags](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 4: So beschreiben Sie Snapshots anhand des Alters

Im folgenden `describe-snapshots` Beispiel werden JMESPath Ausdrücke verwendet, um alle Schnappschüsse zu beschreiben, die von Ihrem AWS Konto vor dem angegebenen Datum erstellt wurden. Es wird nur der Snapshot IDs angezeigt.

```
aws ec2 describe-snapshots \  
  --owner-ids 012345678910 \  
  --query "Snapshots[?(StartTime<='2020-03-31')].[SnapshotId]"
```

Weitere Beispiele für die Verwendung von Filtern finden Sie unter [Ressourcen auflisten und filtern](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 5: So zeigen Sie nur archivierte Snapshots an

Im folgenden `describe-snapshots`-Beispiel werden ausschließlich Snapshots aufgeführt, die auf der Archivstufe gespeichert sind.

```
aws ec2 describe-snapshots \  
  --filters "Name=storage-tier,Values=archive"
```

Ausgabe:

```
{  
  "Snapshots": [  
    {  
      "Description": "Snap A",  
      "Encrypted": false,  
      "VolumeId": "vol-01234567890aaaaaa",  
      "State": "completed",  
      "VolumeSize": 8,  
      "StartTime": "2021-09-07T21:00:00.000Z",  
      "Progress": "100%",  
      "OwnerId": "123456789012",  
      "SnapshotId": "snap-01234567890aaaaaa",  
      "StorageTier": "archive",  
      "Tags": []  
    },  
  ]  
}
```

Weitere Informationen finden Sie unter [Archivierte Snapshots anzeigen](#) im Benutzerhandbuch für Amazon Elastic Compute Cloud.

- API-Einheiten finden Sie [DescribeSnapshots](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt den angegebenen Snapshot.

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

Ausgabe:

```
DataEncryptionKeyId :  
Description          : Created by CreateImage(i-1a2b3c4d) for ami-12345678 from  
vol-12345678  
Encrypted           : False  
KmsKeyId            :  
OwnerAlias          :  
OwnerId             : 123456789012  
Progress            : 100%  
SnapshotId          : snap-12345678  
StartTime           : 10/23/2014 6:01:28 AM  
State               : completed  
StateMessage        :  
Tags                : {}  
VolumeId            : vol-12345678  
VolumeSize          : 8
```

Beispiel 2: Dieses Beispiel beschreibt die Snapshots mit dem Tag „Name“.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

Beispiel 3: Dieses Beispiel beschreibt die Schnappschüsse, die ein 'Name' -Tag mit dem Wert " TestValue haben.

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and  
$_.Tags.Value -eq "TestValue" }
```

Beispiel 4: Dieses Beispiel beschreibt all Ihre Schnappschüsse.

```
Get-EC2Snapshot -Owner self
```

- API-Einheiten finden Sie unter [DescribeSnapshots AWS Tools for PowerShell](#) Cmdlet-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Zeigt den Status eines Snapshots an.

```
async fn show_state(client: &Client, id: &str) -> Result<(), Error> {
    let resp = client
        .describe_snapshots()
        .filters(Filter::builder().name("snapshot-id").values(id).build())
        .send()
        .await?;

    println!(
        "State: {}",
        resp.snapshots().first().unwrap().state().unwrap().as_ref()
    );

    Ok(())
}
```

```
async fn show_snapshots(client: &Client) -> Result<(), Error> {
    // "self" represents your account ID.
    // You can list the snapshots for any account by replacing
    // "self" with that account ID.
    let resp = client.describe_snapshots().owner_ids("self").send().await?;
```



```
let snapshots = resp.snapshots();
let length = snapshots.len();

for snapshot in snapshots {
    println!(
        "ID:          {}",
        snapshot.snapshot_id().unwrap_or_default()
    );
    println!(
        "Description: {}",
        snapshot.description().unwrap_or_default()
    );
    println!("State:          {}", snapshot.state().unwrap().as_ref());
    println!();
}

println!();
println!("Found {} snapshot(s)", length);
println!();

Ok(())
}
```

- API-Einheiten finden Sie [DescribeSnapshots](#) in der AWS SDK-API-Rust-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeSpotDatafeedSubscription** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung von `DescribeSpotDatafeedSubscription`.

CLI

AWS CLI

Um das Spot-Instance-Datenfeed-Abonnement für ein Konto zu beschreiben

Dieser Beispielbefehl beschreibt den Datenfeed für das Konto.

Befehl:

```
aws ec2 describe-spot-datafeed-subscription
```

Ausgabe:

```
{
  "SpotDatafeedSubscription": {
    "OwnerId": "123456789012",
    "Prefix": "spotdata",
    "Bucket": "my-s3-bucket",
    "State": "Active"
  }
}
```

- API-Einheiten finden Sie [DescribeSpotDatafeedSubscription](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt Ihren Spot-Instance-Datenfeed.

```
Get-EC2SpotDatafeedSubscription
```

Ausgabe:

```
Bucket   : my-s3-bucket
Fault    :
OwnerId  : 123456789012
Prefix   : spotdata
State    : Active
```

- API-Einheiten finden Sie unter [DescribeSpotDatafeedSubscription AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie `DescribeSpotFleetInstances` mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeSpotFleetInstances`.

CLI

AWS CLI

Um die Spot-Instances zu beschreiben, die einer Spot-Flotte zugeordnet sind

Dieser Beispielbefehl listet die Spot-Instances auf, die der angegebenen Spot-Flotte zugeordnet sind.

Befehl:

```
aws ec2 describe-spot-fleet-instances --spot-fleet-request-id sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

Ausgabe:

```
{
  "ActiveInstances": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "InstanceType": "m3.medium",
      "SpotInstanceRequestId": "sir-08b93456"
    },
    ...
  ],
  "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE"
}
```

- API-Einheiten finden Sie [DescribeSpotFleetInstances](#) unter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Instances beschrieben, die mit der angegebenen Spot-Flottenanforderung verknüpft sind.

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

Ausgabe:

InstanceId	InstanceType	SpotInstanceRequestId
i-f089262a	c3.large	sir-12345678
i-7e8b24a4	c3.large	sir-87654321

- API-Einheiten finden Sie unter [DescribeSpotFleetInstances AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeSpotFleetRequestHistory** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeSpotFleetRequestHistory`.

CLI

AWS CLI

Um die Geschichte der Spot-Flotte zu beschreiben

Dieser Beispielbefehl gibt den Verlauf für die angegebene Spot-Flotte ab dem angegebenen Zeitpunkt zurück.

Befehl:

```
aws ec2 describe-spot-fleet-request-history --spot-fleet-request-id sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE --start-time 2015-05-26T00:00:00Z
```

Die folgende Beispielausgabe zeigt die erfolgreichen Starts von zwei Spot-Instances für die Spot-Flotte.

Ausgabe:

```
{
  "HistoryRecords": [
    {
      "Timestamp": "2015-05-26T23:17:20.697Z",
      "EventInformation": {
        "EventSubType": "submitted"
      },
      "EventType": "fleetRequestChange"
    },
    {
      "Timestamp": "2015-05-26T23:17:20.873Z",
      "EventInformation": {
        "EventSubType": "active"
      },
      "EventType": "fleetRequestChange"
    },
    {
      "Timestamp": "2015-05-26T23:21:21.712Z",
      "EventInformation": {
        "InstanceId": "i-1234567890abcdef0",
        "EventSubType": "launched"
      },
      "EventType": "instanceChange"
    },
    {
      "Timestamp": "2015-05-26T23:21:21.816Z",
      "EventInformation": {
        "InstanceId": "i-1234567890abcdef1",
        "EventSubType": "launched"
      },
      "EventType": "instanceChange"
    }
  ],
  "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
  "NextToken": "CpHNsscimcV5oH7bSbub03CI2Qms5+ypNpNm
+53MNlR0YcXAkp0xFlfKf91yVxSExmbtma3awYxMFzNA663ZskT0AhtJ6TCb2Z8bQC2EnZgyELbymtWPfpZ1ZbauV
+P+TfG1WxWWB/Vr5dk5d4LfdgA/DRAHUrYgxzrEXAMPLE=",
  "StartTime": "2015-05-26T00:00:00Z"
}
```

- API-Einheiten finden Sie [DescribeSpotFleetRequestHistory](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt den Verlauf der angegebenen Spot-Flottenanfrage.

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

Ausgabe:

```
HistoryRecords      : {Amazon.EC2.Model.HistoryRecord,
  Amazon.EC2.Model.HistoryRecord...}
LastEvaluatedTime  : 12/26/2015 8:29:11 AM
NextToken          :
SpotFleetRequestId : sfr-088bc5f1-7e7b-451a-bd13-757f10672b93
StartTime          : 12/25/2015 8:00:00 AM
```

```
(Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z).HistoryRecords
```

Ausgabe:

EventInformation	EventType	Timestamp
-----	-----	-----
Amazon.EC2.Model.EventInformation	fleetRequestChange	12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation	fleetRequestChange	12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation	fleetRequestChange	12/26/2015 8:23:33 AM
Amazon.EC2.Model.EventInformation	launched	12/26/2015 8:25:34 AM
Amazon.EC2.Model.EventInformation	launched	12/26/2015 8:25:05 AM

- API-Einheiten finden Sie unter [DescribeSpotFleetRequestHistory AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeSpotFleetRequests** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeSpotFleetRequests`.

CLI

AWS CLI

Um Ihre Spot-Flottenanfragen zu beschreiben

In diesem Beispiel werden alle Ihre Spot-Flottenanfragen beschrieben.

Befehl:

```
aws ec2 describe-spot-fleet-requests
```

Ausgabe:

```
{
  "SpotFleetRequestConfigs": [
    {
      "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
      "SpotFleetRequestConfig": {
        "TargetCapacity": 20,
        "LaunchSpecifications": [
          {
            "EbsOptimized": false,
            "NetworkInterfaces": [
              {
                "SubnetId": "subnet-a61dafcf",
                "DeviceIndex": 0,
                "DeleteOnTermination": false,
                "AssociatePublicIpAddress": true,
                "SecondaryPrivateIpAddressCount": 0
              }
            ],
            "InstanceType": "cc2.8xlarge",
            "ImageId": "ami-1a2b3c4d"
          },
          {
            "EbsOptimized": false,
            "NetworkInterfaces": [
              {
                "SubnetId": "subnet-a61dafcf",
                "DeviceIndex": 0,
                "DeleteOnTermination": false,
                "AssociatePublicIpAddress": true,
                "SecondaryPrivateIpAddressCount": 0
              }
            ]
          }
        ]
      }
    }
  ]
}
```

```

        }
      ],
      "InstanceType": "r3.8xlarge",
      "ImageId": "ami-1a2b3c4d"
    }
  ],
  "SpotPrice": "0.05",
  "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role"
},
"SpotFleetRequestState": "active"
},
{
  "SpotFleetRequestId": "sfr-306341ed-9739-402e-881b-ce47bEXAMPLE",
  "SpotFleetRequestConfig": {
    "TargetCapacity": 20,
    "LaunchSpecifications": [
      {
        "EbsOptimized": false,
        "NetworkInterfaces": [
          {
            "SubnetId": "subnet-6e7f829e",
            "DeviceIndex": 0,
            "DeleteOnTermination": false,
            "AssociatePublicIpAddress": true,
            "SecondaryPrivateIpAddressCount": 0
          }
        ],
        "InstanceType": "m3.medium",
        "ImageId": "ami-1a2b3c4d"
      }
    ],
    "SpotPrice": "0.05",
    "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role"
  },
  "SpotFleetRequestState": "active"
}
]
}

```

Um eine Spot-Flottenanfrage zu beschreiben

Dieses Beispiel beschreibt die angegebene Spot-Flottenanfrage.

Befehl:


```
aws ec2 describe-spot-fleet-requests --spot-fleet-request-ids sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

Ausgabe:

```
{
  "SpotFleetRequestConfigs": [
    {
      "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
      "SpotFleetRequestConfig": {
        "TargetCapacity": 20,
        "LaunchSpecifications": [
          {
            "EbsOptimized": false,
            "NetworkInterfaces": [
              {
                "SubnetId": "subnet-a61dafcf",
                "DeviceIndex": 0,
                "DeleteOnTermination": false,
                "AssociatePublicIpAddress": true,
                "SecondaryPrivateIpAddressCount": 0
              }
            ],
            "InstanceType": "cc2.8xlarge",
            "ImageId": "ami-1a2b3c4d"
          },
          {
            "EbsOptimized": false,
            "NetworkInterfaces": [
              {
                "SubnetId": "subnet-a61dafcf",
                "DeviceIndex": 0,
                "DeleteOnTermination": false,
                "AssociatePublicIpAddress": true,
                "SecondaryPrivateIpAddressCount": 0
              }
            ],
            "InstanceType": "r3.8xlarge",
            "ImageId": "ami-1a2b3c4d"
          }
        ],
        "SpotPrice": "0.05",
        "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role"
      }
    }
  ]
}
```

```
    },
    "SpotFleetRequestState": "active"
  }
]
}
```

- API-Einheiten finden Sie [DescribeSpotFleetRequests](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene Spot-Flottenanfrage.

```
Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE | format-list
```

Ausgabe:

```
ConfigData           : Amazon.EC2.Model.SpotFleetRequestConfigData
CreateTime           : 12/26/2015 8:23:33 AM
SpotFleetRequestId   : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
SpotFleetRequestState : active
```

Beispiel 2: Dieses Beispiel beschreibt alle Ihre Spot-Flottenanfragen.

```
Get-EC2SpotFleetRequest
```

- API-Einheiten finden Sie unter [DescribeSpotFleetRequests AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeSpotInstanceRequests** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeSpotInstanceRequests`.

CLI

AWS CLI

Beispiel 1: Um eine Spot-Instance-Anfrage zu beschreiben

Das folgende `describe-spot-instance-requests` Beispiel beschreibt die angegebene Spot-Instance-Anfrage.

```
aws ec2 describe-spot-instance-requests \  
  --spot-instance-request-ids sir-08b93456
```

Ausgabe:

```
{  
  "SpotInstanceRequests": [  
    {  
      "CreateTime": "2018-04-30T18:14:55.000Z",  
      "InstanceId": "i-1234567890abcdef1",  
      "LaunchSpecification": {  
        "InstanceType": "t2.micro",  
        "ImageId": "ami-003634241a8fcdec0",  
        "KeyName": "my-key-pair",  
        "SecurityGroups": [  
          {  
            "GroupName": "default",  
            "GroupId": "sg-e38f24a7"  
          }  
        ],  
        "BlockDeviceMappings": [  
          {  
            "DeviceName": "/dev/sda1",  
            "Ebs": {  
              "DeleteOnTermination": true,  
              "SnapshotId": "snap-0e54a519c999adbbd",  
              "VolumeSize": 8,  
              "VolumeType": "standard",  
              "Encrypted": false  
            }  
          }  
        ],  
        "NetworkInterfaces": [  
          {
```

```

        "DeleteOnTermination": true,
        "DeviceIndex": 0,
        "SubnetId": "subnet-049df61146c4d7901"
    }
  ],
  "Placement": {
    "AvailabilityZone": "us-east-2b",
    "Tenancy": "default"
  },
  "Monitoring": {
    "Enabled": false
  }
},
"LaunchedAvailabilityZone": "us-east-2b",
"ProductDescription": "Linux/UNIX",
"SpotInstanceRequestId": "sir-08b93456",
"SpotPrice": "0.010000"
"State": "active",
"Status": {
  "Code": "fulfilled",
  "Message": "Your Spot request is fulfilled.",
  "UpdateTime": "2018-04-30T18:16:21.000Z"
},
"Tags": [],
"Type": "one-time",
"InstanceInterruptionBehavior": "terminate"
}
]
}

```

Beispiel 2: Um Spot-Instance-Anfragen auf der Grundlage von Filtern zu beschreiben

Im folgenden `describe-spot-instance-requests` Beispiel werden Filter verwendet, um die Ergebnisse auf Spot-Instance-Anfragen mit dem angegebenen Instance-Typ in der angegebenen Availability Zone zu beschränken. Im Beispiel wird der `--query` Parameter verwendet, um nur die Instance anzuzeigenIDs.

```

aws ec2 describe-spot-instance-requests \
  --filters Name=launch.instance-type,Values=m3.medium Name=launched-
availability-zone,Values=us-east-2a \
  --query "SpotInstanceRequests[*].[InstanceId]" \
  --output text

```

Ausgabe:

```
i-057750d42936e468a
i-001efd250faaa6ffa
i-027552a73f021f3bd
...
```

Weitere Beispiele für die Verwendung von Filtern finden Sie unter [Auflisten und Filtern Ihrer Ressourcen](#) im Amazon Elastic Compute Cloud-Benutzerhandbuch.

Beispiel 3: Zur Beschreibung von Spot-Instance-Anfragen auf der Grundlage von Tags

Im folgenden `describe-spot-instance-requests` Beispiel werden Tagfilter verwendet, um die Ergebnisse auf Spot-Instance-Anfragen zu beschränken, die das Tag `cost-center=cc123` enthalten.

```
aws ec2 describe-spot-instance-requests \
  --filters Name=tag:cost-center,Values=cc123
```

Ein Beispiel für die Ausgabe von `describe-spot-instance-requests` finden Sie in Beispiel 1.

Weitere Beispiele für die Verwendung von Tag-Filtern finden Sie unter [Arbeiten mit Tags](#) im EC2Amazon-Benutzerhandbuch.

- API-Einheiten finden Sie [DescribeSpotInstanceRequests](#) in der AWS CLI Befehlsreferenz.

PowerShell**Tools für PowerShell**

Beispiel 1: Dieses Beispiel beschreibt die angegebene Spot-Instance-Anfrage.

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

Ausgabe:

```
ActualBlockHourlyPrice      :
AvailabilityZoneGroup       :
BlockDurationMinutes        : 0
CreateTime                  : 4/8/2015 2:51:33 PM
Fault                       :
```

```

InstanceId           : i-12345678
LaunchedAvailabilityZone : us-west-2b
LaunchGroup          :
LaunchSpecification  : Amazon.EC2.Model.LaunchSpecification
ProductDescription    : Linux/UNIX
SpotInstanceRequestId : sir-12345678
SpotPrice             : 0.020000
State                 : active
Status                : Amazon.EC2.Model.SpotInstanceStatus
Tags                  : {Name}
Type                  : one-time

```

Beispiel 2: Dieses Beispiel beschreibt alle Ihre Spot-Instance-Anfragen.

```
Get-EC2SpotInstanceRequest
```

- API-Einheiten finden Sie unter [DescribeSpotInstanceRequests AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeSpotPriceHistory** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeSpotPriceHistory`.

CLI

AWS CLI

Um die Entwicklung der Spot-Preise zu beschreiben

Dieser Beispielbefehl gibt den Spot-Preisverlauf für `m1.xlarge`-Instances für einen bestimmten Tag im Januar zurück.

Befehl:

```
aws ec2 describe-spot-price-history --instance-types m1.xlarge --start-time 2014-01-06T07:08:09 --end-time 2014-01-06T08:09:10
```

Ausgabe:

```
{
  "SpotPriceHistory": [
    {
      "Timestamp": "2014-01-06T07:10:55.000Z",
      "ProductDescription": "SUSE Linux",
      "InstanceType": "m1.xlarge",
      "SpotPrice": "0.087000",
      "AvailabilityZone": "us-west-1b"
    },
    {
      "Timestamp": "2014-01-06T07:10:55.000Z",
      "ProductDescription": "SUSE Linux",
      "InstanceType": "m1.xlarge",
      "SpotPrice": "0.087000",
      "AvailabilityZone": "us-west-1c"
    },
    {
      "Timestamp": "2014-01-06T05:42:36.000Z",
      "ProductDescription": "SUSE Linux (Amazon VPC)",
      "InstanceType": "m1.xlarge",
      "SpotPrice": "0.087000",
      "AvailabilityZone": "us-west-1a"
    },
    ...
  ]
}
```

Um die Spot-Preisentwicklung für Linux/Amazon UNIX zu beschreiben VPC

Dieser Beispielbefehl gibt den Spot-Preisverlauf für m1.xlarge-, UNIX VPC Linux/Amazon-Instances für einen bestimmten Tag im Januar zurück.

Befehl:

```
aws ec2 describe-spot-price-history --instance-types m1.xlarge --product-
description "Linux/UNIX (Amazon VPC)" --start-time 2014-01-06T07:08:09 --end-
time 2014-01-06T08:09:10
```

Ausgabe:

```
{
  "SpotPriceHistory": [
    {
```

```

    "Timestamp": "2014-01-06T04:32:53.000Z",
    "ProductDescription": "Linux/UNIX (Amazon VPC)",
    "InstanceType": "m1.xlarge",
    "SpotPrice": "0.080000",
    "AvailabilityZone": "us-west-1a"
  },
  {
    "Timestamp": "2014-01-05T11:28:26.000Z",
    "ProductDescription": "Linux/UNIX (Amazon VPC)",
    "InstanceType": "m1.xlarge",
    "SpotPrice": "0.080000",
    "AvailabilityZone": "us-west-1c"
  }
]
}

```

- API-Einheiten finden Sie in der Befehlsreferenz. [DescribeSpotPriceHistory](#) AWS CLI

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die letzten 10 Einträge in der Spot-Preishistorie für den angegebenen Instance-Typ und die Availability Zone abgerufen. Beachten Sie, dass der für den AvailabilityZone Parameter - angegebene Wert für den Regionswert gültig sein muss, der entweder an den Parameter -Region des Cmdlets übergeben wurde (im Beispiel nicht gezeigt) oder als Standard in der Shell festgelegt wurde. Bei diesem Beispielbefehl wird davon ausgegangen, dass die Standardregion 'us-west-2' in der Umgebung festgelegt wurde.

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -
MaxResult 10
```

Ausgabe:

```

AvailabilityZone    : us-west-2a
InstanceType       : c3.large
Price              : 0.017300
ProductDescription  : Linux/UNIX (Amazon VPC)
Timestamp          : 12/25/2015 7:39:49 AM

AvailabilityZone    : us-west-2a

```



```

InstanceType      : c3.large
Price             : 0.017200
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp         : 12/25/2015 7:38:29 AM

AvailabilityZone  : us-west-2a
InstanceType      : c3.large
Price             : 0.017300
ProductDescription : Linux/UNIX (Amazon VPC)
Timestamp         : 12/25/2015 6:57:13 AM
...

```

- API-Einheiten finden Sie unter [DescribeSpotPriceHistory AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeSubnets** mit einem AWS SDK oder CLI.

Die folgenden Codebeispiele zeigen, wie man es benutzt `DescribeSubnets`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erstellen und Verwalten eines ausfallsicheren Services](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

/// <summary>
/// Get all the subnets for a Vpc in a set of availability zones.

```

```
/// </summary>
/// <param name="vpcId">The Id of the Vpc.</param>
/// <param name="availabilityZones">The list of availability zones.</param>
/// <returns>The collection of subnet objects.</returns>
public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
{
    try
    {
        var subnets = new List<Subnet>();
        var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
            new DescribeSubnetsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new("vpc-id", new List<string>() { vpcId }),
                    new("availability-zone", availabilityZones),
                    new("default-for-az", new List<string>() { "true" })
                }
            });

        // Get the entire list using the paginator.
        await foreach (var subnet in subnetPaginator.Subnets)
        {
            subnets.Add(subnet);
        }

        return subnets;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
        {
            _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"An error occurred while describing the
subnets.: {ex.Message}");
        throw;
    }
}
```

```
    }
  }
}
```

- API-Einheiten finden Sie [DescribeSubnets](#) unter AWS SDK for .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So beschreiben Sie Ihre Subnetze

Im folgenden `describe-subnets`-Beispiel werden die Details zu Ihren Subnetzen angezeigt.

```
aws ec2 describe-subnets
```

Ausgabe:

```
{
  "Subnets": [
    {
      "AvailabilityZone": "us-east-1d",
      "AvailabilityZoneId": "use1-az2",
      "AvailableIpAddressCount": 4089,
      "CidrBlock": "172.31.80.0/20",
      "DefaultForAz": true,
      "MapPublicIpOnLaunch": false,
      "MapCustomerOwnedIpOnLaunch": true,
      "State": "available",
      "SubnetId": "subnet-0bb1c79de3EXAMPLE",
      "VpcId": "vpc-0ee975135dEXAMPLE",
      "OwnerId": "111122223333",
      "AssignIpv6AddressOnCreation": false,
      "Ipv6CidrBlockAssociationSet": [],
      "CustomerOwnedIpv4Pool": "pool-2EXAMPLE",
      "SubnetArn": "arn:aws:ec2:us-east-2:111122223333:subnet/subnet-0bb1c79de3EXAMPLE",
      "EnableDns64": false,
      "Ipv6Native": false,
      "PrivateDnsNameOptionsOnLaunch": {
        "HostnameType": "ip-name",
        "EnableResourceNameDnsARecord": false,

```

```

        "EnableResourceNameDnsAAAARecord": false
    }
},
{
    "AvailabilityZone": "us-east-1d",
    "AvailabilityZoneId": "use1-az2",
    "AvailableIpAddressCount": 4089,
    "CidrBlock": "172.31.80.0/20",
    "DefaultForAz": true,
    "MapPublicIpOnLaunch": true,
    "MapCustomerOwnedIpOnLaunch": false,
    "State": "available",
    "SubnetId": "subnet-8EXAMPLE",
    "VpcId": "vpc-3EXAMPLE",
    "OwnerId": "1111222233333",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [],
    "Tags": [
        {
            "Key": "Name",
            "Value": "MySubnet"
        }
    ],
    "SubnetArn": "arn:aws:ec2:us-east-1:111122223333:subnet/
subnet-8EXAMPLE",
    "EnableDns64": false,
    "Ipv6Native": false,
    "PrivateDnsNameOptionsOnLaunch": {
        "HostnameType": "ip-name",
        "EnableResourceNameDnsARecord": false,
        "EnableResourceNameDnsAAAARecord": false
    }
}
]
}

```

Weitere Informationen finden Sie unter [Arbeiten mit VPCs und Subnetzen](#) im AWS VPCBenutzerhandbuch.

Beispiel 2: Zur Beschreibung der Subnetze eines bestimmten VPC

Das folgende `describe-subnets` Beispiel verwendet einen Filter, um Details für die Subnetze der angegebenen Subnetze abzurufen. VPC

```
aws ec2 describe-subnets \  
  --filters "Name=vpc-id,Values=vpc-3EXAMPLE"
```

Ausgabe:

```
{  
  "Subnets": [  
    {  
      "AvailabilityZone": "us-east-1d",  
      "AvailabilityZoneId": "use1-az2",  
      "AvailableIpAddressCount": 4089,  
      "CidrBlock": "172.31.80.0/20",  
      "DefaultForAz": true,  
      "MapPublicIpOnLaunch": true,  
      "MapCustomerOwnedIpOnLaunch": false,  
      "State": "available",  
      "SubnetId": "subnet-8EXAMPLE",  
      "VpcId": "vpc-3EXAMPLE",  
      "OwnerId": "1111222233333",  
      "AssignIpv6AddressOnCreation": false,  
      "Ipv6CidrBlockAssociationSet": [],  
      "Tags": [  
        {  
          "Key": "Name",  
          "Value": "MySubnet"  
        }  
      ],  
      "SubnetArn": "arn:aws:ec2:us-east-1:111122223333:subnet/  
subnet-8EXAMPLE",  
      "EnableDns64": false,  
      "Ipv6Native": false,  
      "PrivateDnsNameOptionsOnLaunch": {  
        "HostnameType": "ip-name",  
        "EnableResourceNameDnsARecord": false,  
        "EnableResourceNameDnsAAAARecord": false  
      }  
    }  
  ]  
}
```

Weitere Informationen finden Sie unter [Arbeiten mit VPCs und Subnetzen](#) im AWS VPCBenutzerhandbuch.

Beispiel 3: So beschreiben Sie die Subnetze mit einem bestimmten Tag

Das folgende `describe-subnets` Beispiel verwendet einen Filter, um die Details dieser Subnetze mit dem Tag `CostCenter=123` und dem `--query` Parameter abzurufen, um das Subnetz IDs der Subnetze mit diesem Tag anzuzeigen.

```
aws ec2 describe-subnets \  
  --filters "Name=tag:CostCenter,Values=123" \  
  --query "Subnets[*].SubnetId" \  
  --output text
```

Ausgabe:

```
subnet-0987a87c8b37348ef  
subnet-02a95061c45f372ee  
subnet-03f720e7de2788d73
```

Weitere Informationen finden Sie unter [Arbeiten mit VPCs und Subnetzen](#) im VPCAmazon-Benutzerhandbuch.

- API-Einheiten finden Sie [DescribeSubnets](#) in der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
const client = new EC2Client({});  
const { Subnets } = await client.send(  
  new DescribeSubnetsCommand({  
    Filters: [  
      { Name: "vpc-id", Values: [state.defaultVpc] },  
      { Name: "availability-zone", Values: state.availabilityZoneNames },  
      { Name: "default-for-az", Values: ["true"] },
```

```
    ],  
  }},  
);
```

- API-Einheiten finden Sie [DescribeSubnets](#) unter AWS SDK for JavaScript API-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das angegebene Subnetz.

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

Ausgabe:

```
AvailabilityZone      : us-west-2c  
AvailableIpAddressCount : 251  
CidrBlock             : 10.0.0.0/24  
DefaultForAz         : False  
MapPublicIpOnLaunch  : False  
State                 : available  
SubnetId              : subnet-1a2b3c4d  
Tags                  : {}  
VpcId                 : vpc-12345678
```

Beispiel 2: Dieses Beispiel beschreibt alle Ihre Subnetze.

```
Get-EC2Subnet
```

- API-Einheiten finden Sie unter [DescribeSubnets AWS Tools for PowerShell](#) Cmdlet-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
```



```
self.ssm_client = ssm_client
self.iam_client = iam_client
sts_client = boto3.client("sts")
self.account_id = sts_client.get_caller_identity()["Account"]

self.key_pair_name = f"{resource_prefix}-key-pair"
self.launch_template_name = f"{resource_prefix}-template-"
self.group_name = f"{resource_prefix}-group"

# Happy path
self.instance_policy_name = f"{resource_prefix}-pol"
self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"

# Failure mode
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
    """
    Gets the default subnets in a VPC for a specified list of Availability
    Zones.

    :param vpc_id: The ID of the VPC to look up.
    :param zones: The list of Availability Zones to look up.
    :return: The list of subnets found.
    """
    # Ensure that 'zones' is a list, even if None is passed
    if zones is None:
        zones = []
    try:
        paginator = self.ec2_client.get_paginator("describe_subnets")
        page_iterator = paginator.paginate(
            Filters=[
                {"Name": "vpc-id", "Values": [vpc_id]},
                {"Name": "availability-zone", "Values": zones},
                {"Name": "default-for-az", "Values": ["true"]},
            ]
        )

        subnets = []
```

```

    for page in page_iterator:
        subnets.extend(page["Subnets"])

    log.info("Found %s subnets for the specified zones.", len(subnets))
    return subnets
except ClientError as err:
    log.error(
        f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
{zones}."
    )
    error_code = err.response["Error"]["Code"]
    if error_code == "InvalidVpcID.NotFound":
        log.error(
            "The specified VPC ID does not exist. "
            "Please check the VPC ID and try again."
        )
    # Add more error-specific handling as needed
    log.error(f"Full error:\n\t{err}")

```

- API-Einheiten finden Sie unter [DescribeSubnets](#) Python (Boto3) API -Referenz. AWS SDK

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeTags** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeTags`.

CLI

AWS CLI

Beispiel 1: Um alle Tags für eine einzelne Ressource zu beschreiben

Das folgende `describe-tags` Beispiel beschreibt die Tags für die angegebene Instanz.

```
aws ec2 describe-tags \
  --filters "Name=resource-id,Values=i-1234567890abcdef8"
```

Ausgabe:

```
{
  "Tags": [
    {
      "ResourceType": "instance",
      "ResourceId": "i-1234567890abcdef8",
      "Value": "Test",
      "Key": "Stack"
    },
    {
      "ResourceType": "instance",
      "ResourceId": "i-1234567890abcdef8",
      "Value": "Beta Server",
      "Key": "Name"
    }
  ]
}
```

Beispiel 2: Um alle Tags für einen Ressourcentyp zu beschreiben

Das folgende `describe-tags` Beispiel beschreibt die Tags für Ihre Volumes.

```
aws ec2 describe-tags \
  --filters "Name=resource-type,Values=volume"
```

Ausgabe:

```
{
  "Tags": [
    {
      "ResourceType": "volume",
      "ResourceId": "vol-1234567890abcdef0",
      "Value": "Project1",
      "Key": "Purpose"
    },
    {
      "ResourceType": "volume",
      "ResourceId": "vol-049df61146c4d7901",
      "Value": "Logs",
      "Key": "Purpose"
    }
  ]
}
```

```
]
}
```

Beispiel 3: Um all Ihre Tags zu beschreiben

Das folgende `describe-tags` Beispiel beschreibt die Tags für all Ihre Ressourcen.

```
aws ec2 describe-tags
```

Beispiel 4: Um die Tags für Ihre Ressourcen anhand eines Tag-Schlüssels zu beschreiben

Das folgende `describe-tags` Beispiel beschreibt die Tags für Ihre Ressourcen, die ein Tag mit dem Schlüssel `Stack` haben.

```
aws ec2 describe-tags \
  --filters Name=key,Values=Stack
```

Ausgabe:

```
{
  "Tags": [
    {
      "ResourceType": "volume",
      "ResourceId": "vol-027552a73f021f3b",
      "Value": "Production",
      "Key": "Stack"
    },
    {
      "ResourceType": "instance",
      "ResourceId": "i-1234567890abcdef8",
      "Value": "Test",
      "Key": "Stack"
    }
  ]
}
```

Beispiel 5: Um die Tags für Ihre Ressourcen auf der Grundlage eines Tag-Schlüssels und eines Tag-Werts zu beschreiben

Im folgenden `describe-tags` Beispiel werden die Tags für Ihre Ressourcen beschrieben, die das Tag `Stack=Test` enthalten.

```
aws ec2 describe-tags \
  --filters Name=key,Values=Stack Name=value,Values=Test
```

Ausgabe:

```
{
  "Tags": [
    {
      "ResourceType": "image",
      "ResourceId": "ami-3ac336533f021f3bd",
      "Value": "Test",
      "Key": "Stack"
    },
    {
      "ResourceType": "instance",
      "ResourceId": "i-1234567890abcdef8",
      "Value": "Test",
      "Key": "Stack"
    }
  ]
}
```

Im folgenden `describe-tags` Beispiel wird eine alternative Syntax verwendet, um Ressourcen mit dem Tag zu beschreiben `Stack=Test`.

```
aws ec2 describe-tags \
  --filters "Name=tag:Stack,Values=Test"
```

Das folgende `describe-tags` Beispiel beschreibt die Tags für all Ihre Instances, die ein Tag mit dem Schlüssel `Purpose` und ohne Wert haben.

```
aws ec2 describe-tags \
  --filters "Name=resource-type,Values=instance" "Name=key,Values=Purpose" "Name=value,Values="
```

Ausgabe:

```
{
  "Tags": [
    {
      "ResourceType": "instance",
```

```

        "ResourceId": "i-1234567890abcdef5",
        "Value": null,
        "Key": "Purpose"
    }
]
}

```

- API-Einheiten finden Sie [DescribeTags](#) unter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Tags für den Ressourcentyp 'image' abgerufen

```
Get-EC2Tag -Filter @{Name="resource-type";Values="image"}
```

Ausgabe:

Key	ResourceId	ResourceType	Value
Name	ami-0a123b4ccb567a8ea	image	Win7-Imported
auto-delete	ami-0a123b4ccb567a8ea	image	never

Beispiel 2: In diesem Beispiel werden alle Tags für alle Ressourcen abgerufen und nach Ressourcentyp gruppiert

```
Get-EC2Tag | Group-Object resourcetype
```

Ausgabe:

Count	Name	Group
9	subnet	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription...}
53	instance	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription...}
3	route-table	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}

```

5 security-group      {Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription...}
30 volume            {Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription...}
1 internet-gateway   {Amazon.EC2.Model.TagDescription}
3 network-interface  {Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
4 elastic-ip         {Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription}
1 dhcp-options       {Amazon.EC2.Model.TagDescription}
2 image              {Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription}
3 vpc                {Amazon.EC2.Model.TagDescription,
Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}

```

Beispiel 3: In diesem Beispiel werden alle Ressourcen mit dem Tag 'auto-delete' und dem Wert 'no' für die angegebene Region angezeigt

```
Get-EC2Tag -Region eu-west-1 -Filter @{"Name"="tag:auto-delete";Values="no"}
```

Ausgabe:

Key	ResourceId	ResourceType	Value
---	-----	-----	-----
auto-delete	i-0f1bce234d5dd678b	instance	no
auto-delete	vol-01d234aa5678901a2	volume	no
auto-delete	vol-01234bfb5def6f7b8	volume	no
auto-delete	vol-01ccb23f4c5e67890	volume	no

Beispiel 4: In diesem Beispiel werden alle Ressourcen mit dem Tag 'auto-delete' mit dem Wert 'no' und weitere Filter in der nächsten Pipe abgerufen, um nur die Ressourcentypen 'Instanz' zu analysieren, und erstellt schließlich das Tag 'ThisInstance' für jede Instanzressource, wobei der Wert die Instanz-ID selbst ist

```

Get-EC2Tag -Region eu-west-1 -Filter @{"Name"="tag:auto-delete";Values="no"}
| Where-Object ResourceType -eq "instance" | ForEach-Object {New-EC2Tag -
ResourceId $_.ResourceId -Tag @{"Key"="ThisInstance";Value=$_.ResourceId}}

```

Beispiel 5: In diesem Beispiel werden Tags für alle Instanzressourcen sowie Namensschlüssel abgerufen und in einem Tabellenformat angezeigt

```
Get-EC2Tag -Filter @{{Name="resource-
type";Values="instance"},@{Name="key";Values="Name"}} | Select-Object ResourceId,
@{Name="Name-Tag";Expression={$PSItem.Value}} | Format-Table -AutoSize
```

Ausgabe:

ResourceId	Name-Tag
i-012e3cb4df567e1aa	jump1
i-01c23a45d6fc7a89f	repro-3

- APIEinzelheiten finden Sie unter [DescribeTags AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeVolumeAttribute** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeVolumeAttribute`.

CLI

AWS CLI

Um ein Volumenattribut zu beschreiben

Dieser Beispielbefehl beschreibt das `autoEnableIo` Attribut des Volumes mit der ID `vol-049df61146c4d7901`.

Befehl:

```
aws ec2 describe-volume-attribute --volume-id vol-049df61146c4d7901 --
attribute autoEnableIO
```

Ausgabe:


```
{
  "AutoEnableIO": {
    "Value": false
  },
  "VolumeId": "vol-049df61146c4d7901"
}
```

- API-Einheiten finden Sie [DescribeVolumeAttribute](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das angegebene Attribut des angegebenen Volumes.

```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

Ausgabe:

AutoEnableIO	ProductCodes	VolumeId
False	{}	vol-12345678

- API-Einheiten finden Sie unter [DescribeVolumeAttribute AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeVolumeStatus** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeVolumeStatus`.

CLI

AWS CLI

Um den Status eines einzelnen Volumes zu beschreiben

Dieser Beispielbefehl beschreibt den Status des Volumes `vol-1234567890abcdef0`.

Befehl:

```
aws ec2 describe-volume-status --volume-ids vol-1234567890abcdef0
```

Ausgabe:

```
{
  "VolumeStatuses": [
    {
      "VolumeStatus": {
        "Status": "ok",
        "Details": [
          {
            "Status": "passed",
            "Name": "io-enabled"
          },
          {
            "Status": "not-applicable",
            "Name": "io-performance"
          }
        ]
      },
      "AvailabilityZone": "us-east-1a",
      "VolumeId": "vol-1234567890abcdef0",
      "Actions": [],
      "Events": []
    }
  ]
}
```

Um den Status von kaputten Volumes zu beschreiben

Dieser Beispielbefehl beschreibt den Status aller Datenträger, die beeinträchtigt sind. In dieser Beispielausgabe gibt es keine beeinträchtigten Volumes.

Befehl:

```
aws ec2 describe-volume-status --filters Name=volume-  
status.status,Values=impaired
```

Ausgabe:

```
{
  "VolumeStatuses": []
}
```

Wenn Sie ein Volume haben, bei dem die Statusüberprüfung nicht bestanden wurde (der Status ist beeinträchtigt), finden Sie weitere Informationen unter Arbeiten mit einem beeinträchtigten Volumen im EC2Amazon-Benutzerhandbuch.

- API-Einheiten finden Sie [DescribeVolumeStatus](#) in der AWS CLI Befehlsreferenz.

PowerShell**Tools für PowerShell**

Beispiel 1: Dieses Beispiel beschreibt den Status des angegebenen Volumes.

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

Ausgabe:

```
Actions           : {}
AvailabilityZone  : us-west-2a
Events           : {}
VolumeId         : vol-12345678
VolumeStatus     : Amazon.EC2.Model.VolumeStatusInfo
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

Ausgabe:

Details	Status
-----	-----
{io-enabled, io-performance}	ok

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

Ausgabe:

Name	Status
----	-----
io-enabled	passed
io-performance	not-applicable

- API-Einheiten finden Sie unter [DescribeVolumeStatus AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeVolumes** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeVolumes`.

CLI

AWS CLI

Beispiel 1: Um einen Band zu beschreiben

Das folgende `describe-volumes` Beispiel beschreibt die angegebenen Volumes in der aktuellen Region.

```
aws ec2 describe-volumes \
  --volume-ids vol-049df61146c4d7901 vol-1234567890abcdef0
```

Ausgabe:

```
{
  "Volumes": [
    {
      "AvailabilityZone": "us-east-1a",
      "Attachments": [
        {
          "AttachTime": "2013-12-18T22:35:00.000Z",
          "InstanceId": "i-1234567890abcdef0",
          "VolumeId": "vol-049df61146c4d7901",
          "State": "attached",
          "DeleteOnTermination": true,

```

```

        "Device": "/dev/sda1"
      }
    ],
    "Encrypted": true,
    "KmsKeyId": "arn:aws:kms:us-east-2a:123456789012:key/8c5b2c63-
b9bc-45a3-a87a-5513eEXAMPLE",
    "VolumeType": "gp2",
    "VolumeId": "vol-049df61146c4d7901",
    "State": "in-use",
    "Iops": 100,
    "SnapshotId": "snap-1234567890abcdef0",
    "CreateTime": "2019-12-18T22:35:00.084Z",
    "Size": 8
  },
  {
    "AvailabilityZone": "us-east-1a",
    "Attachments": [],
    "Encrypted": false,
    "VolumeType": "gp2",
    "VolumeId": "vol-1234567890abcdef0",
    "State": "available",
    "Iops": 300,
    "SnapshotId": "",
    "CreateTime": "2020-02-27T00:02:41.791Z",
    "Size": 100
  }
]
}

```

Beispiel 2: Zur Beschreibung von Volumes, die an eine bestimmte Instanz angehängt sind

Das folgende `describe-volumes` Beispiel beschreibt alle Volumes, die sowohl an die angegebene Instance angehängt als auch so eingestellt sind, dass sie gelöscht werden, wenn die Instance beendet wird.

```

aws ec2 describe-volumes \
  --region us-east-1 \
  --filters Name=attachment.instance-
id,Values=i-1234567890abcdef0 Name=attachment.delete-on-termination,Values=true

```

Ein Beispiel für die Ausgabe von `describe-volumes` finden Sie in Beispiel 1.

Beispiel 3: Um verfügbare Volumes in einer bestimmten Availability Zone zu beschreiben

Im folgenden `describe-volumes` Beispiel werden alle Volumes beschrieben, die den Status `available` haben und sich in der angegebenen Availability Zone befinden.

```
aws ec2 describe-volumes \  
  --filters Name=status,Values=available Name=availability-zone,Values=us-east-1a
```

Ein Beispiel für die Ausgabe von `describe-volumes` finden Sie in Beispiel 1.

Beispiel 4: Um Volumes anhand von Tags zu beschreiben

Das folgende `describe-volumes` Beispiel beschreibt alle Volumes, die den Tag-Schlüssel `Name` und einen Wert haben, der mit `beginntest` beginnt. Die Ausgabe wird dann mit einer Abfrage gefiltert, bei der nur die Tags und IDs der Volumes angezeigt werden.

```
aws ec2 describe-volumes \  
  --filters Name=tag:Name,Values=Test* \  
  --query "Volumes[*].{ID:VolumeId,Tag:Tags}"
```

Ausgabe:

```
[  
  {  
    "Tag": [  
      {  
        "Value": "Test2",  
        "Key": "Name"  
      }  
    ],  
    "ID": "vol-1234567890abcdef0"  
  },  
  {  
    "Tag": [  
      {  
        "Value": "Test1",  
        "Key": "Name"  
      }  
    ],  
    "ID": "vol-049df61146c4d7901"  
  }  
]
```

Weitere Beispiele für die Verwendung von Tag-Filtern finden Sie unter [Arbeiten mit Tags](#) im EC2Amazon-Benutzerhandbuch.

- API-Einheiten finden Sie [DescribeVolumes](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das angegebene EBS Volumen.

```
Get-EC2Volume -VolumeId vol-12345678
```

Ausgabe:

```
Attachments      : {}
AvailabilityZone  : us-west-2c
CreateTime       : 7/17/2015 4:35:19 PM
Encrypted        : False
Iops             : 90
KmsKeyId         :
Size            : 30
SnapshotId      : snap-12345678
State           : in-use
Tags            : {}
VolumeId        : vol-12345678
VolumeType      : standard
```

Beispiel 2: Dieses Beispiel beschreibt Ihre EBS Volumes, die den Status „verfügbar“ haben.

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

Ausgabe:

```
Attachments      : {}
AvailabilityZone  : us-west-2c
CreateTime       : 12/21/2015 2:31:29 PM
Encrypted        : False
Iops             : 60
KmsKeyId         :
Size            : 20
```

```

SnapshotId      : snap-12345678
State           : available
Tags            : {}
VolumeId        : vol-12345678
VolumeType      : gp2
...

```

Beispiel 3: Dieses Beispiel beschreibt alle Ihre EBS Volumes.

```
Get-EC2Volume
```

- API-Einheiten finden Sie unter [DescribeVolumes AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeVpcAttribute** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeVpcAttribute`.

CLI

AWS CLI

Um das `enableDnsSupport` Attribut zu beschreiben

Dieses Beispiel beschreibt das `enableDnsSupport` Attribut. Dieses Attribut gibt an, ob die DNS-Auflösung für aktiviert ist VPC. Wenn dieses Attribut zutrifft `true`, löst der DNS-Amazon-Server DNS-Hostnamen für Ihre Instanzen in die entsprechenden IP-Adressen auf; andernfalls nicht.

Befehl:

```
aws ec2 describe-vpc-attribute --vpc-id vpc-a01106c2 --attribute enableDnsSupport
```

Ausgabe:

```
{
```



```
"VpcId": "vpc-a01106c2",
"EnableDnsSupport": {
  "Value": true
}
}
```

Um das Attribut zu beschreiben `enableDnsHostnames`

Dieses Beispiel beschreibt das `enableDnsHostnames` Attribut. Dieses Attribut gibt an, ob die Instances im VPC Get DNS Hostnames gestartet wurden. Wenn dieses Attribut zutrifft `true`, VPC erhalten die Instanzen in der DNS Liste Hostnamen, andernfalls nicht.

Befehl:

```
aws ec2 describe-vpc-attribute --vpc-id vpc-a01106c2 --
attribute enableDnsHostnames
```

Ausgabe:

```
{
  "VpcId": "vpc-a01106c2",
  "EnableDnsHostnames": {
    "Value": true
  }
}
```

- API-Einheiten finden Sie [DescribeVpcAttribute](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Attribut 'enableDnsSupport' beschrieben.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

Ausgabe:

```
EnableDnsSupport
```

```
-----  
True
```

Beispiel 2: In diesem Beispiel wird das Attribut 'enableDnsHostnames' beschrieben.

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

Ausgabe:

```
EnableDnsHostnames  
-----  
True
```

- API-Einheiten finden Sie unter [DescribeVpcAttribute AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeVpcClassicLink** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeVpcClassicLink`.

CLI

AWS CLI

Um den ClassicLink-Status Ihrer zu beschreiben VPCs

Dieses Beispiel listet den ClassicLink-Status von `vpc-88888888` auf.

Befehl:

```
aws ec2 describe-vpc-classic-link --vpc-id vpc-88888888
```

Ausgabe:

```
{
```

```

"Vpcs": [
  {
    "ClassicLinkEnabled": true,
    "VpcId": "vpc-88888888",
    "Tags": [
      {
        "Value": "classiclinkvpc",
        "Key": "Name"
      }
    ]
  }
]
}

```

In diesem Beispiel werden nur diejenigen aufgeführten VPCs, die für Classiclink aktiviert sind (der Filterwert von ist auf gesetzt). `is-classic-link-enabled` `true`

Befehl:

```
aws ec2 describe-vpc-classic-link --filter "Name=is-classic-link-enabled,Values=true"
```

- API Einheiten finden Sie unter [DescribeVpcClassicLink AWS CLI Befehlsreferenz](#).

PowerShell

Tools für PowerShell

Beispiel 1: Das obige Beispiel gibt alle VPCs mit ihrem ClassicLinkEnabled Bundesstaat für die Region zurück

```
Get-EC2VpcClassicLink -Region eu-west-1
```

Ausgabe:

```

ClassicLinkEnabled Tags    VpcId
-----
False              {Name} vpc-0fc1ff23f45b678eb
False              {}      vpc-01e23c4a5d6db78e9
False              {Name} vpc-0123456b078b9d01f

```

```
False      {}      vpc-12cf3b4f
False      {Name} vpc-0b12d3456a7e8901d
```

- API-Einheiten finden Sie unter [DescribeVpcClassicLink AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeVpcClassicLinkDnsSupport** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeVpcClassicLinkDnsSupport`.

CLI

AWS CLI

Um die ClassicLink DNS-Unterstützung für Ihre zu beschreibenden VPCs

Dieses Beispiel beschreibt den ClassicLink DNS-Support-Status aller Ihrer VPCs.

Befehl:

```
aws ec2 describe-vpc-classic-link-dns-support
```

Ausgabe:

```
{
  "Vpcs": [
    {
      "VpcId": "vpc-88888888",
      "ClassicLinkDnsSupported": true
    },
    {
      "VpcId": "vpc-1a2b3c4d",
      "ClassicLinkDnsSupported": false
    }
  ]
}
```

- API-Einheiten finden Sie [DescribeVpcClassicLinkDnsSupport](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt den ClassicLink DNS Unterstützungstatus von VPCs für die Region eu-west-1

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

Ausgabe:

```
ClassicLinkDnsSupported VpcId
-----
False                   vpc-0b12d3456a7e8910d
False                   vpc-12cf3b4f
```

- API-Einheiten finden Sie unter [DescribeVpcClassicLinkDnsSupport AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeVpcEndpointServices** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeVpcEndpointServices`.

CLI

AWS CLI

Beispiel 1: Um alle VPC Endpunktdienste zu beschreiben

Das folgende "describe-vpc-endpoint-services" Beispiel listet alle VPC Endpunktdienste für eine AWS Region auf.

```
aws ec2 describe-vpc-endpoint-services
```

Ausgabe:

```
{
  "ServiceDetails": [
    {
      "ServiceType": [
        {
          "ServiceType": "Gateway"
        }
      ],
      "AcceptanceRequired": false,
      "ServiceName": "com.amazonaws.us-east-1.dynamodb",
      "VpcEndpointPolicySupported": true,
      "Owner": "amazon",
      "AvailabilityZones": [
        "us-east-1a",
        "us-east-1b",
        "us-east-1c",
        "us-east-1d",
        "us-east-1e",
        "us-east-1f"
      ],
      "BaseEndpointDnsNames": [
        "dynamodb.us-east-1.amazonaws.com"
      ]
    },
    {
      "ServiceType": [
        {
          "ServiceType": "Interface"
        }
      ],
      "PrivateDnsName": "ec2.us-east-1.amazonaws.com",
      "ServiceName": "com.amazonaws.us-east-1.ec2",
      "VpcEndpointPolicySupported": false,
      "Owner": "amazon",
      "AvailabilityZones": [
        "us-east-1a",
        "us-east-1b",
        "us-east-1c",
        "us-east-1d",
        "us-east-1e",
        "us-east-1f"
      ]
    }
  ],
}
```

```

    "AcceptanceRequired": false,
    "BaseEndpointDnsNames": [
      "ec2.us-east-1.vpce.amazonaws.com"
    ]
  },
  {
    "ServiceType": [
      {
        "ServiceType": "Interface"
      }
    ],
    "PrivateDnsName": "ssm.us-east-1.amazonaws.com",
    "ServiceName": "com.amazonaws.us-east-1.ssm",
    "VpcEndpointPolicySupported": true,
    "Owner": "amazon",
    "AvailabilityZones": [
      "us-east-1a",
      "us-east-1b",
      "us-east-1c",
      "us-east-1d",
      "us-east-1e"
    ],
    "AcceptanceRequired": false,
    "BaseEndpointDnsNames": [
      "ssm.us-east-1.vpce.amazonaws.com"
    ]
  }
],
"ServiceNames": [
  "com.amazonaws.us-east-1.dynamodb",
  "com.amazonaws.us-east-1.ec2",
  "com.amazonaws.us-east-1.ec2messages",
  "com.amazonaws.us-east-1.elasticloadbalancing",
  "com.amazonaws.us-east-1.kinesis-streams",
  "com.amazonaws.us-east-1.s3",
  "com.amazonaws.us-east-1.ssm"
]
}

```

Weitere Informationen finden Sie unter [Verfügbare AWS Dienstnamen anzeigen](#) im Benutzerhandbuch für AWS PrivateLink.

Beispiel 2: Um die Details zu einem Endpunktdienst zu beschreiben

Das folgende Beispiel `describe-vpc-endpoint-services` listet die Details des Amazon S3 S3-Schnittstellen-Endpunktdienstes auf

```
aws ec2 describe-vpc-endpoint-services \
  --filter "Name=service-type,Values=Interface" Name=service-
  name,Values=com.amazonaws.us-east-1.s3
```

Ausgabe:

```
{
  "ServiceDetails": [
    {
      "ServiceName": "com.amazonaws.us-east-1.s3",
      "ServiceId": "vpce-svc-081d84efcdEXAMPLE",
      "ServiceType": [
        {
          "ServiceType": "Interface"
        }
      ],
      "AvailabilityZones": [
        "us-east-1a",
        "us-east-1b",
        "us-east-1c",
        "us-east-1d",
        "us-east-1e",
        "us-east-1f"
      ],
      "Owner": "amazon",
      "BaseEndpointDnsNames": [
        "s3.us-east-1.vpce.amazonaws.com"
      ],
      "VpcEndpointPolicySupported": true,
      "AcceptanceRequired": false,
      "ManagesVpcEndpoints": false,
      "Tags": []
    }
  ],
  "ServiceNames": [
    "com.amazonaws.us-east-1.s3"
  ]
}
```


Weitere Informationen finden Sie unter [Verfügbare AWS Servicenamen anzeigen](#) im Benutzerhandbuch für AWS PrivateLink.

- API-Einheiten finden Sie [DescribeVpcEndpointServices](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt den EC2 VPC Endpunkt-Service mit dem angegebenen Filter, in diesem Fall `com.amazonaws.eu-west-1.ecs`. Außerdem wird die Eigenschaft erweitert und die Details werden angezeigt `ServiceDetails`

```
Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{"Name"="service-name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty ServiceDetails
```

Ausgabe:

```
AcceptanceRequired      : False
AvailabilityZones        : {eu-west-1a, eu-west-1b, eu-west-1c}
BaseEndpointDnsNames    : {ecs.eu-west-1.vpce.amazonaws.com}
Owner                   : amazon
PrivateDnsName          : ecs.eu-west-1.amazonaws.com
ServiceName             : com.amazonaws.eu-west-1.ecs
ServiceType             : {Amazon.EC2.Model.ServiceTypeDetail}
VpcEndpointPolicySupported : False
```

Beispiel 2: In diesem Beispiel werden alle EC2 VPC Endpunktdienste abgerufen und das `ServiceNames` passende „`ssm`“ zurückgegeben

```
Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty Servicenames | Where-Object { -match "ssm" }
```

Ausgabe:

```
com.amazonaws.eu-west-1.ssm
com.amazonaws.eu-west-1.ssmmessages
```

- API-Einheiten finden Sie unter [DescribeVpcEndpointServices AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeVpcEndpoints** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeVpcEndpoints`.

CLI

AWS CLI

Um Ihre VPC Endpunkte zu beschreiben

Im folgenden `describe-vpc-endpoints` Beispiel werden Details zu all Ihren VPC Endpunkten angezeigt.

```
aws ec2 describe-vpc-endpoints
```

Ausgabe:

```
{
  "VpcEndpoints": [
    {
      "PolicyDocument": "{\"Version\":\"2008-10-17\",\"Statement\":\n\n[[{\"Effect\":\"Allow\",\"Principal\":\"*\",\"Action\":\"*\",\"Resource\":\"*\n\n\"}]]}\",
      "VpcId": "vpc-aabb1122",
      "NetworkInterfaceIds": [],
      "SubnetIds": [],
      "PrivateDnsEnabled": true,
      "State": "available",
      "ServiceName": "com.amazonaws.us-east-1.dynamodb",
      "RouteTableIds": [
        "rtb-3d560345"
      ],
      "Groups": [],
      "VpcEndpointId": "vpce-032a826a",
      "VpcEndpointType": "Gateway",
      "CreationTimestamp": "2017-09-05T20:41:28Z",
      "DnsEntries": [],
      "OwnerId": "123456789012"
    },
  ],
}
```

```

    {
      "PolicyDocument": "{\n  \"Statement\": [\n    {\n      \"Action\":\n        \"*\", \n      \"Effect\": \"Allow\", \n      \"Principal\": \"*\", \n      \"Resource\": \"*\"\n    }]\n  }",
      "VpcId": "vpc-1a2b3c4d",
      "NetworkInterfaceIds": [
        "eni-2ec2b084",
        "eni-1b4a65cf"
      ],
      "SubnetIds": [
        "subnet-d6fcaa8d",
        "subnet-7b16de0c"
      ],
      "PrivateDnsEnabled": false,
      "State": "available",
      "ServiceName": "com.amazonaws.us-east-1.elasticloadbalancing",
      "RouteTableIds": [],
      "Groups": [
        {
          "GroupName": "default",
          "GroupId": "sg-54e8bf31"
        }
      ],
      "VpcEndpointId": "vpce-0f89a33420c1931d7",
      "VpcEndpointType": "Interface",
      "CreationTimestamp": "2017-09-05T17:55:27.583Z",
      "DnsEntries": [
        {
          "HostedZoneId": "Z7HUB22UULQXV",
          "DnsName": "vpce-0f89a33420c1931d7-
bluzidnv.elasticloadbalancing.us-east-1.vpce.amazonaws.com"
        },
        {
          "HostedZoneId": "Z7HUB22UULQXV",
          "DnsName": "vpce-0f89a33420c1931d7-bluzidnv-us-
east-1b.elasticloadbalancing.us-east-1.vpce.amazonaws.com"
        },
        {
          "HostedZoneId": "Z7HUB22UULQXV",
          "DnsName": "vpce-0f89a33420c1931d7-bluzidnv-us-
east-1a.elasticloadbalancing.us-east-1.vpce.amazonaws.com"
        }
      ],
      "OwnerId": "123456789012"
    }
  
```

```

    },
    {
      "VpcEndpointId": "vpce-aabbaabbaabbaabba",
      "VpcEndpointType": "GatewayLoadBalancer",
      "VpcId": "vpc-111122223333aabb",
      "ServiceName": "com.amazonaws.vpce.us-east-1.vpce-
svc-123123a1c43abc123",
      "State": "available",
      "SubnetIds": [
        "subnet-0011aabbcc2233445"
      ],
      "RequesterManaged": false,
      "NetworkInterfaceIds": [
        "eni-01010120203030405"
      ],
      "CreationTimestamp": "2020-11-11T08:06:03.522Z",
      "Tags": [],
      "OwnerId": "123456789012"
    }
  ]
}

```

Weitere Informationen finden Sie unter [VPC Endpoints](#) im VPC Amazon-Benutzerhandbuch.

- API Einzelheiten finden Sie [DescribeVpcEndpoints](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt einen oder mehrere Ihrer VPC Endpunkte für die Region eu-west-1. Anschließend wird die Ausgabe an den nächsten Befehl weitergeleitet, der die VpcEndpointId Eigenschaft auswählt und die VPC Array-ID als String-Array zurückgibt

```

Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty
VpcEndpointId

```

Ausgabe:

```

vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
vpce-0012e34d567890e12

```

```
vpce-0c123db4567890123
```

Beispiel 2: Dieses Beispiel beschreibt alle VPC-Endpunkte für die Region eu-west-1 und wählt VpcEndpointId,, ServiceName und PrivateDnsEnabled Eigenschaften aus VpcId, um sie in einem tabellarischen Format darzustellen

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
  ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

Ausgabe:

VpcEndpointId	VpcId	ServiceName
vpce-02a2ab2f2f2cc2f2d	vpc-0fc6ff46f65b039eb	com.amazonaws.eu-west-1.ssm
PrivateDnsEnabled		
True		
vpce-01d1b111a1114561b	vpc-0fc6ff46f65b039eb	com.amazonaws.eu-west-1.ec2
PrivateDnsEnabled		
True		
vpce-0011e23d45167e838	vpc-0fc6ff46f65b039eb	com.amazonaws.eu-west-1.ec2messages
PrivateDnsEnabled		
True		
vpce-0c123db4567890123	vpc-0fc6ff46f65b039eb	com.amazonaws.eu-west-1.ssmmessages
PrivateDnsEnabled		
True		

Beispiel 3: In diesem Beispiel wird das Richtliniendokument für den VPC Endpunkt vpce-01a2ab3f4f5cc6f7d in eine JSON-Datei exportiert

```
Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d |
  Select-Object -expand PolicyDocument | Out-File vpce_policyDocument.json
```

- Einzelheiten [DescribeVpcEndpoints AWS Tools for PowerShell](#) finden Sie unter Cmdlet-Referenz. API

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter. [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeVpcs** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutztDescribeVpcs.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erstellen und Verwalten eines ausfallsicheren Services](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Get the default VPC for the account.
/// </summary>
/// <returns>The default VPC object.</returns>
public async Task<Vpc> GetDefaultVpc()
{
    try
    {
        var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
            new DescribeVpcsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new("is-default", new List<string>() { "true" })
                }
            });
        return vpcResponse.Vpcs[0];
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "UnauthorizedOperation")
        {
            _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
        }
    }
}
```

```

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"An error occurred while describing the vpcs.:
{ex.Message}");
        throw;
    }
}

```

- API-Einheiten finden Sie [DescribeVpcs](#) unter AWS SDK for .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: Um alle Ihre zu beschreiben VPCs

Im folgenden `describe-vpcs` Beispiel werden Details zu Ihren VPCs abgerufen.

```
aws ec2 describe-vpcs
```

Ausgabe:

```

{
  "Vpcs": [
    {
      "CidrBlock": "30.1.0.0/16",
      "DhcpOptionsId": "dopt-19edf471",
      "State": "available",
      "VpcId": "vpc-0e9801d129EXAMPLE",
      "OwnerId": "111122223333",
      "InstanceTenancy": "default",
      "CidrBlockAssociationSet": [
        {
          "AssociationId": "vpc-cidr-assoc-062c64cfafEXAMPLE",
          "CidrBlock": "30.1.0.0/16",
          "CidrBlockState": {
            "State": "associated"
          }
        }
      ]
    }
  ]
}

```

```

    ],
    "IsDefault": false,
    "Tags": [
      {
        "Key": "Name",
        "Value": "Not Shared"
      }
    ]
  },
  {
    "CidrBlock": "10.0.0.0/16",
    "DhcpOptionsId": "dopt-19edf471",
    "State": "available",
    "VpcId": "vpc-06e4ab6c6cEXAMPLE",
    "OwnerId": "222222222222",
    "InstanceTenancy": "default",
    "CidrBlockAssociationSet": [
      {
        "AssociationId": "vpc-cidr-assoc-00b17b4eddEXAMPLE",
        "CidrBlock": "10.0.0.0/16",
        "CidrBlockState": {
          "State": "associated"
        }
      }
    ]
  },
  "IsDefault": false,
  "Tags": [
    {
      "Key": "Name",
      "Value": "Shared VPC"
    }
  ]
}
]
}

```

Beispiel 2: Um ein bestimmtes zu beschreiben VPC

Im folgenden `describe-vpcs` Beispiel werden Details für das angegebene VPC Objekt abgerufen.

```

aws ec2 describe-vpcs \
  --vpc-ids vpc-06e4ab6c6cEXAMPLE

```



Ausgabe:

```
{
  "Vpcs": [
    {
      "CidrBlock": "10.0.0.0/16",
      "DhcpOptionsId": "dopt-19edf471",
      "State": "available",
      "VpcId": "vpc-06e4ab6c6cEXAMPLE",
      "OwnerId": "111122223333",
      "InstanceTenancy": "default",
      "CidrBlockAssociationSet": [
        {
          "AssociationId": "vpc-cidr-assoc-00b17b4eddEXAMPLE",
          "CidrBlock": "10.0.0.0/16",
          "CidrBlockState": {
            "State": "associated"
          }
        }
      ],
      "IsDefault": false,
      "Tags": [
        {
          "Key": "Name",
          "Value": "Shared VPC"
        }
      ]
    }
  ]
}
```

- API-Einheiten finden Sie unter [DescribeVpcs AWS CLI](#) Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
const client = new EC2Client({});
const { Vpcs } = await client.send(
  new DescribeVpcsCommand({
    Filters: [{ Name: "is-default", Values: ["true"] }],
  }),
);
```

- API-Einheiten finden Sie [DescribeVpcs](#) unter AWS SDK for JavaScript API-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebenen VPC.

```
Get-EC2Vpc -VpcId vpc-12345678
```

Ausgabe:

```
CidrBlock      : 10.0.0.0/16
DhcpOptionsId  : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault      : False
State          : available
Tags           : {Name}
VpcId          : vpc-12345678
```

Beispiel 2: Dieses Beispiel beschreibt die Standardeinstellung VPC (es kann nur eine pro Region geben). Wenn Ihr Konto EC2 -Classic in dieser Region unterstützt, gibt es keinen StandardVPC.

```
Get-EC2Vpc -Filter @{"Name"="isDefault"; Values="true"}
```

Ausgabe:

```
CidrBlock      : 172.31.0.0/16
DhcpOptionsId  : dopt-12345678
InstanceTenancy : default
IsDefault      : True
```

```
State      : available
Tags       : {}
VpcId      : vpc-45678901
```

Beispiel 3: Dieses Beispiel beschreibt die VPCs, die dem angegebenen Filter entsprechen (d. CIDR h. einen haben, der dem Wert '10.0.0.0/16' entspricht und sich im Status 'verfügbar' befinden).

```
Get-EC2Vpc -Filter @{Name="cidr";
  Values="10.0.0.0/16"},@{Name="state";Values="available"}
```

Beispiel 4: Dieses Beispiel beschreibt alle Ihre VPCs

```
Get-EC2Vpc
```

- API-Einheiten finden Sie unter [DescribeVpcs AWS Tools for PowerShell](#) Cmdlet-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
```

```

        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.

```

```

"""
try:
    response = self.ec2_client.describe_vpcs(
        Filters=[{"Name": "is-default", "Values": ["true"]}])
except ClientError as err:
    error_code = err.response["Error"]["Code"]
    log.error("Failed to retrieve the default VPC.")
    if error_code == "UnauthorizedOperation":
        log.error(
            "You do not have the necessary permissions to describe VPCs.
            "
            "Ensure that your AWS IAM user or role has the correct
permissions."
        )
    elif error_code == "InvalidParameterValue":
        log.error(
            "One or more parameters are invalid. Check the request
parameters."
        )

    log.error(f"Full error:\n\t{err}")
else:
    if "Vpcs" in response and response["Vpcs"]:
        log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
        return response["Vpcs"][0]
    else:
        pass

```

- API-Einheiten finden Sie unter [DescribeVpcs](#) Python (Boto3) API -Referenz. AWS SDK

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeVpnConnections** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeVpnConnections`.

CLI

AWS CLI

Beispiel 1: Um Ihre VPN Verbindungen zu beschreiben

Das folgende `describe-vpn-connections` Beispiel beschreibt alle Ihre Site-to-Site VPN Verbindungen.

```
aws ec2 describe-vpn-connections
```

Ausgabe:

```
{
  "VpnConnections": [
    {
      "CustomerGatewayConfiguration": "...configuration information...",
      "CustomerGatewayId": "cgw-01234567abcde1234",
      "Category": "VPN",
      "State": "available",
      "Type": "ipsec.1",
      "VpnConnectionId": "vpn-1122334455aabbccd",
      "TransitGatewayId": "tgw-00112233445566aab",
      "Options": {
        "EnableAcceleration": false,
        "StaticRoutesOnly": true,
        "LocalIpv4NetworkCidr": "0.0.0.0/0",
        "RemoteIpv4NetworkCidr": "0.0.0.0/0",
        "TunnelInsideIpVersion": "ipv4"
      },
      "Routes": [],
      "Tags": [
        {
          "Key": "Name",
          "Value": "CanadaVPN"
        }
      ],
      "VgwTelemetry": [
        {
          "AcceptedRouteCount": 0,
          "LastStatusChange": "2020-07-29T10:35:11.000Z",
          "OutsideIpAddress": "203.0.113.3",
          "Status": "DOWN",

```

```

        "StatusMessage": ""
      },
      {
        "AcceptedRouteCount": 0,
        "LastStatusChange": "2020-09-02T09:09:33.000Z",
        "OutsideIpAddress": "203.0.113.5",
        "Status": "UP",
        "StatusMessage": ""
      }
    ]
  }
]
}

```

Weitere Informationen finden Sie im AWS Site-to-Site VPNBenutzerhandbuch unter [So AWS Site-to-Site VPN funktioniert](#) es.

Beispiel 2: Um Ihre verfügbaren VPN Verbindungen zu beschreiben

Das folgende `describe-vpn-connections` Beispiel beschreibt Ihre Site-to-Site VPN Verbindungen mit dem Status `available`.

```

aws ec2 describe-vpn-connections \
  --filters "Name=state,Values=available"

```

Weitere Informationen finden Sie im AWS Site-to-Site VPNBenutzerhandbuch unter [So AWS Site-to-Site VPN funktioniert](#) es.

- API-Einheiten finden Sie [DescribeVpnConnections](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt die angegebene VPN Verbindung.

```

Get-EC2VpnConnection -VpnConnectionId vpn-12345678

```

Ausgabe:

```

CustomerGatewayConfiguration : [XML document]

```

```

CustomerGatewayId      : cgw-1a2b3c4d
Options                 : Amazon.EC2.Model.VpnConnectionOptions
Routes                 : {Amazon.EC2.Model.VpnStaticRoute}
State                  : available
Tags                   : {}
Type                   : ipsec.1
VgwTelemetry           : {Amazon.EC2.Model.VgwTelemetry,
  Amazon.EC2.Model.VgwTelemetry}
VpnConnectionId       : vpn-12345678
VpnGatewayId          : vgw-1a2b3c4d

```

Beispiel 2: Dieses Beispiel beschreibt jede VPN Verbindung, deren Status entweder ausstehend oder verfügbar ist.

```

$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnConnection -Filter $filter

```

Beispiel 3: Dieses Beispiel beschreibt all Ihre VPN Verbindungen.

```
Get-EC2VpnConnection
```

- APIEinzelheiten finden Sie unter [DescribeVpnConnections AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DescribeVpnGateways** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DescribeVpnGateways`.

CLI

AWS CLI

Um Ihre virtuellen privaten Gateways zu beschreiben

Dieses Beispiel beschreibt Ihre virtuellen privaten Gateways.

Befehl:

```
aws ec2 describe-vpn-gateways
```

Ausgabe:

```
{
  "VpnGateways": [
    {
      "State": "available",
      "Type": "ipsec.1",
      "VpnGatewayId": "vgw-f211f09b",
      "VpcAttachments": [
        {
          "State": "attached",
          "VpcId": "vpc-98eb5ef5"
        }
      ]
    },
    {
      "State": "available",
      "Type": "ipsec.1",
      "VpnGatewayId": "vgw-9a4cacf3",
      "VpcAttachments": [
        {
          "State": "attaching",
          "VpcId": "vpc-a01106c2"
        }
      ]
    }
  ]
}
```

- API-Einheiten finden Sie [DescribeVpnGateways](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel beschreibt das angegebene Virtual Private Gateway.

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

Ausgabe:

```
AvailabilityZone :  
State           : available  
Tags            : {}  
Type            : ipsec.1  
VpcAttachments : {vpc-12345678}  
VpnGatewayId   : vgw-1a2b3c4d
```

Beispiel 2: Dieses Beispiel beschreibt jedes virtuelle private Gateway, dessen Status entweder ausstehend oder verfügbar ist.

```
$filter = New-Object Amazon.EC2.Model.Filter  
$filter.Name = "state"  
$filter.Values = @( "pending", "available" )  
  
Get-EC2VpnGateway -Filter $filter
```

Beispiel 3: Dieses Beispiel beschreibt alle Ihre virtuellen privaten Gateways.

```
Get-EC2VpnGateway
```

- API-Einheiten finden Sie unter [DescribeVpnGateways AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DetachInternetGateway** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DetachInternetGateway`.

CLI

AWS CLI

Um ein Internet-Gateway von Ihrem zu trennen VPC

Im folgenden `detach-internet-gateway` Beispiel wird das angegebene Internet-Gateway vom spezifischen getrennt. VPC

```
aws ec2 detach-internet-gateway \  
  --internet-gateway-id igw-0d0fb496b3EXAMPLE \  
  --vpc-id vpc-0a60eb65b4EXAMPLE
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Internet-Gateways](#) im VPCAmazon-Benutzerhandbuch.

- APIEinzelheiten finden Sie [DetachInternetGateway](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Internet-Gateway vom angegebenen VPC getrennt.

```
Dismount-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

- APIEinzelheiten finden Sie unter [DetachInternetGateway AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DetachNetworkInterface** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DetachNetworkInterface`.

CLI

AWS CLI

Um eine Netzwerkschnittstelle von Ihrer Instance zu trennen

In diesem Beispiel wird die angegebene Netzwerkschnittstelle von der angegebenen Instanz getrennt. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 detach-network-interface --attachment-id eni-attach-66c4350a
```

- API-Einheiten finden Sie unter [DetachNetworkInterface AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Verbindung zwischen einer Netzwerkschnittstelle und einer Instanz entfernt.

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- API-Einheiten finden Sie unter [DetachNetworkInterface AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DetachVolume** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DetachVolume`.

CLI

AWS CLI

Um ein Volume von einer Instanz zu trennen

Dieser Beispielbefehl trennt das Volume (`vol-049df61146c4d7901`) von der Instance, an die es angehängt ist.

Befehl:

```
aws ec2 detach-volume --volume-id vol-1234567890abcdef0
```

Ausgabe:

```
{
  "AttachTime": "2014-02-27T19:23:06.000Z",
  "InstanceId": "i-1234567890abcdef0",
  "VolumeId": "vol-049df61146c4d7901",
  "State": "detaching",
  "Device": "/dev/sdb"
}
```

- API-Einheiten finden Sie [DetachVolume](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Volume getrennt.

```
Dismount-EC2Volume -VolumeId vol-12345678
```

Ausgabe:

```
AttachTime      : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device          : /dev/sdh
InstanceId      : i-1a2b3c4d
State          : detaching
VolumeId       : vol-12345678
```

Beispiel 2: Sie können auch die Instanz-ID und den Gerätenamen angeben, um sicherzustellen, dass Sie das richtige Volume trennen.

```
Dismount-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

- API-Einheiten finden Sie unter [DetachVolume AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DetachVpnGateway** mit einem CLI

Die folgenden Codebeispiele zeigen die Verwendung `DetachVpnGateway`.

CLI

AWS CLI

Um ein virtuelles privates Gateway von Ihrem zu trennen VPC

In diesem Beispiel wird das angegebene virtuelle private Gateway vom angegebenen getrennt. VPC Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 detach-vpn-gateway --vpn-gateway-id vgw-9a4cacf3 --vpc-id vpc-a01106c2
```

- API-Einheiten finden Sie unter [DetachVpnGateway AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene virtuelle private Gateway vom angegebenen VPC getrennt.

```
Dismount-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

- API-Einheiten finden Sie unter [DetachVpnGateway AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DisableVgwRoutePropagation** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DisableVgwRoutePropagation`.

CLI

AWS CLI

Um die Route-Propagierung zu deaktivieren

In diesem Beispiel wird verhindert, dass das angegebene Virtual Private Gateway statische Routen an die angegebene Routentabelle weitergibt. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 disable-vgw-route-propagation --route-table-id rtb-22574640 --gateway-id vgw-9a4caf3
```

- API-Einheiten finden Sie unter [DisableVgwRoutePropagation AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird verhindert, dass Routen automatisch an die VGW angegebene Routingtabelle weitergegeben werden.

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- API-Einheiten finden Sie unter [DisableVgwRoutePropagation AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DisableVpcClassicLink** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DisableVpcClassicLink`.

CLI

AWS CLI

Zur Deaktivierung ClassicLink für ein VPC

Dieses Beispiel deaktiviert ClassicLink für vpc-88888888.

Befehl:

```
aws ec2 disable-vpc-classic-link --vpc-id vpc-88888888
```

Ausgabe:

```
{
  "Return": true
}
```

- APIEinzelheiten finden Sie in der Befehlsreferenz. [DisableVpcClassicLink](#) AWS CLI

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel deaktiviert EC2VpcClassicLink für den vpc-01e23c4a5d6db78e9. Es gibt entweder True oder False zurück

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- APIEinzelheiten finden Sie unter [DisableVpcClassicLink AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter. [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DisableVpcClassicLinkDnsSupport** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DisableVpcClassicLinkDnsSupport`.

CLI

AWS CLI

Um die ClassicLink DNS Unterstützung für ein zu deaktivieren VPC

In diesem Beispiel wird die ClassicLink DNS Unterstützung für `vpc-88888888` deaktiviert.

Befehl:

```
aws ec2 disable-vpc-classic-link-dns-support --vpc-id vpc-88888888
```

Ausgabe:

```
{
  "Return": true
}
```

- APIEinzelheiten finden Sie [DisableVpcClassicLinkDnsSupport](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die ClassicLink DNS Unterstützung für `vpc-0b12d3456a7e8910d` deaktiviert

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- APIEinzelheiten finden [DisableVpcClassicLinkDnsSupport AWS Tools for PowerShell](#) Sie unter Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DisassociateAddress** mit einem AWS SDK oder CLI


Die folgenden Codebeispiele zeigen, wie man es benutzt `DisassociateAddress`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)

.NET

AWS SDK for .NET

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Disassociate an Elastic IP address from an EC2 instance.
/// </summary>
/// <param name="associationId">The association Id.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DisassociateIp(string associationId)
{
    try
    {
        var response = await _amazonEC2.DisassociateAddressAsync(
            new DisassociateAddressRequest { AssociationId =
associationId });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidAssociationID.NotFound")
        {
            _logger.LogError(
                $"AssociationId is invalid, unable to disassociate address.
{ec2Exception.Message}");
        }

        return false;
    }
    catch (Exception ex)
```

```

    {
        _logger.LogError(
            $"An error occurred while disassociating the Elastic IP.:
{ex.Message}");
        return false;
    }
}

```

- API-Einheiten finden Sie [DisassociateAddress](#) unter AWS SDK for .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

#####
# function ec2_disassociate_address
#
# This function disassociates an Elastic IP address from an Amazon Elastic
# Compute Cloud (Amazon EC2) instance.
#
# Parameters:
#     -a association_id - The association ID that represents the association of
#     the Elastic IP address with an instance.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#
#####
function ec2_disassociate_address() {
    local association_id response

    # Function to display usage information
    function usage() {
        echo "function ec2_disassociate_address"
    }
}

```

```
    echo "Disassociates an Elastic IP address from an Amazon Elastic Compute
Cloud (Amazon EC2) instance."
    echo "  -a association_id - The association ID that represents the
association of the Elastic IP address with an instance."
    echo ""
}

# Parse the command-line arguments
while getopts "a:h" option; do
    case "${option}" in
        a) association_id="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

# Validate the input parameters
if [[ -z "$association_id" ]]; then
    errecho "ERROR: You must provide an association ID with the -a parameter."
    return 1
fi

response=$(aws ec2 disassociate-address \
    --association-id "$association_id") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports disassociate-address operation failed."
    errecho "$response"
    return 1
}

return 0
}
```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```
#####  
# function errecho  
#  
# This function outputs everything sent to it to STDERR (standard error output).  
#####  
function errecho() {  
    printf "%s\n" "$*" 1>&2  
}  
  
#####  
# function aws_cli_error_log()  
#  
# This function is used to log the error messages from the AWS CLI.  
#  
# The function expects the following argument:  
#     $1 - The error code returned by the AWS CLI.  
#  
# Returns:  
#     0: - Success.  
#  
#####  
function aws_cli_error_log() {  
    local err_code=$1  
    errecho "Error code : $err_code"  
    if [ "$err_code" == 1 ]; then  
        errecho " One or more S3 transfers failed."  
    elif [ "$err_code" == 2 ]; then  
        errecho " Command line failed to parse."  
    elif [ "$err_code" == 130 ]; then  
        errecho " Process received SIGINT."  
    elif [ "$err_code" == 252 ]; then  
        errecho " Command syntax invalid."  
    elif [ "$err_code" == 253 ]; then  
        errecho " The system environment or configuration was invalid."  
    elif [ "$err_code" == 254 ]; then  
        errecho " The service returned an error."  
    elif [ "$err_code" == 255 ]; then  
        errecho " 255 is a catch-all error."  
    fi  
  
    return 0  
}
```

- API-Einheiten finden Sie [DisassociateAddress](#) in der AWS CLI Befehlsreferenz.

CLI

AWS CLI

So trennen Sie die Zuordnung von Elastic IP-Adressen in -Classic EC2

In diesem Beispiel wird die Zuordnung einer Elastic IP-Adresse zu einer Instanz in -Classic getrennt. EC2 Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 disassociate-address --public-ip 198.51.100.0
```

Um die Zuordnung einer Elastic IP-Adresse in - zu trennen EC2 VPC

In diesem Beispiel wird die Zuordnung einer Elastic IP-Adresse zu einer Instance in a getrennt. VPC Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 disassociate-address --association-id eipassoc-2bebb745
```

- API-Einheiten finden Sie unter [DisassociateAddress AWS CLI](#) Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**  
 * Disassociates an Elastic IP address from an instance asynchronously.  
 */
```

```

    * @param associationId The ID of the association you want to disassociate.
    * @return a {@link CompletableFuture} representing the asynchronous
operation of disassociating the address. The
    *     {@link CompletableFuture} will complete with a {@link
DisassociateAddressResponse} when the operation is
    *     finished.
    * @throws RuntimeException if the disassociation of the address fails.
    */
    public CompletableFuture<DisassociateAddressResponse>
disassociateAddressAsync(String associationId) {
        Ec2AsyncClient ec2 = getAsyncClient();
        DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
            .associationId(associationId)
            .build();

        // Disassociate the address asynchronously.
        CompletableFuture<DisassociateAddressResponse> response =
ec2.disassociateAddress(addressRequest);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to disassociate address", ex);
            }
        });

        return response;
    }

```

- API-Einheiten finden Sie [DisassociateAddress](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import { DisassociateAddressCommand, EC2Client } from "@aws-sdk/client-ec2";
```

```
/**
 * Disassociate an Elastic IP address from an instance.
 * @param {{ associationId: string }} options
 */
export const main = async ({ associationId }) => {
  const client = new EC2Client({});
  const command = new DisassociateAddressCommand({
    // You can also use PublicIp, but that is for EC2 classic which is being
    // retired.
    AssociationId: associationId,
  });

  try {
    await client.send(command);
    console.log("Successfully disassociated address");
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAssociationID.NotFound"
    ) {
      console.warn(`${caught.message}.`);
    } else {
      throw caught;
    }
  }
};
```

- API-Einheiten finden Sie [DisassociateAddress](#) in der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.


```
suspend fun disassociateAddressSc(associationIdVal: String?) {
    val addressRequest =
        DisassociateAddressRequest {
            associationId = associationIdVal
        }
    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.disassociateAddress(addressRequest)
        println("You successfully disassociated the address!")
    }
}
```

- API-Einheiten finden Sie [DisassociateAddress](#) in der AWS SDK-API-Kotlin-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Verbindung zwischen der angegebenen Elastic IP-Adresse und der angegebenen Instance in a VPC getrennt.

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

Beispiel 2: In diesem Beispiel wird die Verbindung zwischen der angegebenen Elastic IP-Adresse und der angegebenen Instance in EC2 -Classic getrennt.

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- API-Einheiten finden Sie unter [DisassociateAddress AWS Tools for PowerShell Cmdlet](#)-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
    actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
            = None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

        def __init__(self, ec2_client: Any) -> None:
            """
            Initializes the ElasticIpWrapper with an EC2 client.

            :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
            level
                access to AWS EC2 services.
            """
            self.ec2_client = ec2_client
            self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
        client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

```

```
def disassociate(self, allocation_id: str) -> None:
    """
    Removes an association between an Elastic IP address and an instance.
    When the
    association is removed, the instance is assigned a new public IP address.

    :param allocation_id: The allocation ID of the Elastic IP to
    disassociate.
    :raises ClientError: If the disassociation fails, such as when the
    association ID is not found.
    """
    elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
    allocation_id)
    if elastic_ip is None or elastic_ip.instance_id is None:
        logger.info(
            f"No association found for Elastic IP with allocation ID
    {allocation_id}."
        )
        return

    try:
        # Retrieve the association ID before disassociating
        response =
    self.ec2_client.describe_addresses(AllocationIds=[allocation_id])
        association_id = response["Addresses"][0].get("AssociationId")

        if association_id:

    self.ec2_client.disassociate_address(AssociationId=association_id)
        elastic_ip.instance_id = None # Remove the instance association
    else:
        logger.info(
            f"No Association ID found for Elastic IP with allocation ID
    {allocation_id}."
        )

    except ClientError as err:
        if err.response["Error"]["Code"] == "InvalidAssociationID.NotFound":
            logger.error(
                f"Failed to disassociate Elastic IP {allocation_id} "
                "because the specified association ID for the Elastic IP
    address was not found. "
            )
```

```

        "Verify the association ID and ensure the Elastic IP is
        currently associated with a "
        "resource before attempting to disassociate it."
    )
    raise

```

- API-Einheiten finden Sie unter [DisassociateAddress](#) Python (Boto3) API-Referenz. AWS SDK

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

pub async fn disassociate_ip_address(&self, association_id: &str) ->
Result<(), EC2Error> {
    self.client
        .disassociate_address()
        .association_id(association_id)
        .send()
        .await?;
    Ok(())
}

```

- API-Einheiten finden Sie [DisassociateAddress](#) in der AWS SDK API Rust-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **DisassociateRouteTable** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `DisassociateRouteTable`.

CLI

AWS CLI

Um die Zuordnung einer Routentabelle aufzuheben

In diesem Beispiel wird die Verbindung zwischen der angegebenen Routing-Tabelle und dem angegebenen Subnetz getrennt. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 disassociate-route-table --association-id rtbassoc-781d0d1a
```

- API-Einheiten finden Sie unter [DisassociateRouteTable AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Zuordnung zwischen einer Routing-Tabelle und einem Subnetz entfernt.

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- API-Einheiten finden Sie unter [DisassociateRouteTable AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **EnableVgwRoutePropagation** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `EnableVgwRoutePropagation`.

CLI

AWS CLI

Um die Route-Propagierung zu aktivieren

In diesem Beispiel kann das angegebene Virtual Private Gateway statische Routen an die angegebene Routentabelle weitergeben. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 enable-vgw-route-propagation --route-table-id rtb-22574640 --gateway-id vgw-9a4cacf3
```

- API-Einheiten finden Sie unter [EnableVgwRoutePropagation AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene System aktiviert VGW, um Routen automatisch an die angegebene Routingtabelle weiterzugeben.

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- API-Einheiten finden Sie unter [EnableVgwRoutePropagation AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **EnableVolumeIo** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `EnableVolumeIo`.

CLI

AWS CLI

Um I/O für ein Volume zu aktivieren

In diesem Beispiel wird I/O auf dem Volume aktiviert `vol-1234567890abcdef0`.

Befehl:

```
aws ec2 enable-volume-io --volume-id vol-1234567890abcdef0
```

Ausgabe:

```
{  
  "Return": true  
}
```

- API-Einheiten finden Sie [EnableVolumeIo](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden I/O-Operationen für das angegebene Volume aktiviert, wenn I/O-Operationen deaktiviert wurden.

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- API-Einheiten finden Sie unter [EnableVolumeIo AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **EnableVpcClassicLink** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `EnableVpcClassicLink`.

CLI

AWS CLI

Um ein VPC für zu aktivieren ClassicLink

In diesem Beispiel wird vpc-8888888 für aktiviert. ClassicLink

Befehl:

```
aws ec2 enable-vpc-classic-link --vpc-id vpc-88888888
```

Ausgabe:

```
{  
  "Return": true  
}
```

- APIEinzelheiten finden Sie in der Befehlsreferenz [EnableVpcClassicLink](#).AWS CLI

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel aktiviert VPC vpc-0123456b789b0d12f für ClassicLink

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

Ausgabe:

```
True
```

- APIEinzelheiten finden [EnableVpcClassicLink](#)Sie AWS Tools for PowerShell unter Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter. [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **EnableVpcClassicLinkDnsSupport** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `EnableVpcClassicLinkDnsSupport`.

CLI

AWS CLI

Um die ClassicLink DNS Unterstützung für ein zu aktivieren VPC

Dieses Beispiel aktiviert die ClassicLink DNS Unterstützung für `vpc-88888888`.

Befehl:

```
aws ec2 enable-vpc-classic-link-dns-support --vpc-id vpc-88888888
```

Ausgabe:

```
{
  "Return": true
}
```

- API Einzelheiten finden Sie [EnableVpcClassicLinkDnsSupport](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel ermöglicht `vpc-0b12d3456a7e8910d` die Unterstützung der Hostnamenauflösung für DNS ClassicLink

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- API Einzelheiten [EnableVpcClassicLinkDnsSupport](#) finden AWS Tools for PowerShell Sie unter Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **GetConsoleOutput** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `GetConsoleOutput`.

CLI

AWS CLI

Beispiel 1: Um die Konsolenausgabe zu erhalten

Im folgenden `get-console-output` Beispiel wird die Konsolenausgabe für die angegebene Linux-Instanz abgerufen.

```
aws ec2 get-console-output \  
  --instance-id i-1234567890abcdef0
```

Ausgabe:

```
{  
  "InstanceId": "i-1234567890abcdef0",  
  "Timestamp": "2013-07-25T21:23:53.000Z",  
  "Output": "..."  
}
```

Weitere Informationen finden Sie unter [Ausgabe der Instance-Konsole](#) im EC2 Amazon-Benutzerhandbuch.

Beispiel 2: Um die neueste Konsolenausgabe zu erhalten

Im folgenden `get-console-output` Beispiel wird die neueste Konsolenausgabe für die angegebene Linux-Instanz abgerufen.

```
aws ec2 get-console-output \  
  --instance-id i-1234567890abcdef0 \  
  --latest \  
  --output text
```

Ausgabe:

```
i-1234567890abcdef0 [ 0.000000] Command line: root=LABEL=/ console=tty1  
console=ttyS0 selinux=0 nvme_core.io_timeout=4294967295
```

```
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
[ 0.000000] x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'
...
Cloud-init v. 0.7.6 finished at Wed, 09 May 2018 19:01:13 +0000. Datasource DataSourceEc2. Up 21.50 seconds
Amazon Linux AMI release 2018.03
Kernel 4.14.26-46.32.amzn1.x
```

Weitere Informationen finden Sie unter [Ausgabe der Instance-Konsole](#) im EC2Amazon-Benutzerhandbuch.

- API-Einheiten finden Sie [GetConsoleOutput](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Konsolenausgabe für die angegebene Linux-Instance abgerufen. Die Konsolenausgabe ist codiert.

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

Ausgabe:

InstanceId	Output
-----	-----
i-0e194d3c47c123637	WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs

Beispiel 2: In diesem Beispiel wird die codierte Konsolenausgabe in einer Variablen gespeichert und anschließend dekodiert.

```
$Output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Output_encoded))
```

- API-Einheiten finden Sie unter [GetConsoleOutput AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter. [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **GetHostReservationPurchasePreview** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `GetHostReservationPurchasePreview`.

CLI

AWS CLI

Um eine Kaufvorschau für eine Dedicated Host-Reservierung zu erhalten

Dieses Beispiel bietet eine Vorschau der Kosten für eine bestimmte Dedicated Host-Reservierung für den angegebenen Dedicated Host in Ihrem Konto.

Befehl:

```
aws ec2 get-host-reservation-purchase-preview --offering-id hxo-03f707bf363b6b324
--host-id-set h-013abcd2a00cbd123
```

Ausgabe:

```
{
  "TotalHourlyPrice": "1.499",
  "Purchase": [
    {
      "HourlyPrice": "1.499",
      "InstanceFamily": "m4",
      "PaymentOption": "NoUpfront",
      "HostIdSet": [
        "h-013abcd2a00cbd123"
      ],
      "UpfrontPrice": "0.000",
      "Duration": 31536000
    }
  ],
  "TotalUpfrontPrice": "0.000"
}
```

- API-Einheiten finden Sie [GetHostReservationPurchasePreview](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine Vorschau eines Reservierungskaufs mit Konfigurationen angezeigt, die denen Ihres Dedicated Hosts h-01e23f4cd567890f1 entsprechen

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet
h-01e23f4cd567890f1
```

Ausgabe:

```
CurrencyCode Purchase TotalHourlyPrice TotalUpfrontPrice
-----
                {}          1.307             0.000
```

- Einzelheiten [GetHostReservationPurchasePreview AWS Tools for PowerShell](#) finden Sie unter Cmdlet-Referenz. API

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **GetPasswordData** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `GetPasswordData`.

CLI

AWS CLI

Um das verschlüsselte Passwort zu erhalten

In diesem Beispiel wird das verschlüsselte Passwort abgerufen.

Befehl:

```
aws ec2 get-password-data --instance-id i-1234567890abcdef0
```

Ausgabe:

```
{
  "InstanceId": "i-1234567890abcdef0",
  "Timestamp": "2013-08-07T22:18:38.000Z",
  "PasswordData": "gSlJFq+VpcZXqy+iktxMF6NyxQ4qCrT4+ga0uN0enX1MmgXPTj7XEXAMPLE
UQ+YeFfb+L1U4C4AKv652Ux1iRB3CPTYp7WmU3TUnhsuBd+p6LVk7T2lKUm160Xbk6WPW1VYYm/TRPB1
e1DQ7PY4an/DgZT4mwcpRfigzhniQgDDe01InvSDcwoUTwNs0Y1S8ouri2W4n5GNlriM3Q0AnNVe1Vz/
53TkDtxbNoU606M1gK9zUWSxqEgwvbV2j8c5rP0WCuaMWSF14ziDu4bd7q+4RSyi8NUsVWnKZ4aEZffu
DPGzKrF5yL1f3etP2L4ZR6CvG7K1hx7VK0QVN32Dajw=="
}
```

Um das entschlüsselte Passwort zu erhalten

In diesem Beispiel wird das entschlüsselte Passwort abgerufen.

Befehl:

```
aws ec2 get-password-data --instance-id i-1234567890abcdef0 --priv-launch-key C:
\Keys\MyKeyPair.pem
```

Ausgabe:

```
{
  "InstanceId": "i-1234567890abcdef0",
  "Timestamp": "2013-08-30T23:18:05.000Z",
  "PasswordData": "&ViJ652e*u"
}
```

- API-Einheiten finden Sie [GetPasswordData](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.*;
import java.util.concurrent.CompletableFuture;

/**
 * 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 GetPasswordData {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <instanceId>

            Where:
                instanceId - An instance id value that you can obtain from the
AWS Management Console.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        String instanceId = args[0];
```

```
Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
    .region(Region.US_EAST_1)
    .build();

try {
    CompletableFuture<Void> future = getPasswordDataAsync(ec2AsyncClient,
instanceId);
    future.join();
} catch (RuntimeException rte) {
    System.err.println("An exception occurred: " + (rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage()));
}
}

/**
 * Fetches the password data for the specified EC2 instance asynchronously.
 *
 * @param ec2AsyncClient the EC2 asynchronous client to use for the request
 * @param instanceId instanceId the ID of the EC2 instance for which you want
to fetch the password data
 * @return a {@link CompletableFuture} that completes when the password data
has been fetched
 * @throws RuntimeException if there was a failure in fetching the password
data
 */
public static CompletableFuture<Void> getPasswordDataAsync(Ec2AsyncClient
ec2AsyncClient, String instanceId) {
    GetPasswordDataRequest getPasswordDataRequest =
GetPasswordDataRequest.builder()
        .instanceId(instanceId)
        .build();

    CompletableFuture<GetPasswordDataResponse> response =
ec2AsyncClient.getPasswordData(getPasswordDataRequest);
    response.whenComplete((getPasswordDataResponse, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to get password data for
instance: " + instanceId, ex);
        } else if (getPasswordDataResponse == null ||
getPasswordDataResponse.getPasswordData().isEmpty()) {
            throw new RuntimeException("No password data found for instance:
" + instanceId);
        } else {
```



```
        String encryptedPasswordData =
getPasswordDataResponse.passwordData();
        System.out.println("Encrypted Password Data: " +
encryptedPasswordData);
    }
});

return response.thenApply(resp -> null);
}
}
```

- API-Einheiten finden Sie [GetPasswordData](#) in der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Passwort entschlüsselt, das Amazon dem Administratorkonto für die angegebene Windows-Instanz EC2 zugewiesen hat. Da eine PEM-Datei angegeben wurde, wird automatisch die Einstellung des Schalters `-Decrypt` übernommen.

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

Ausgabe:

```
mYZ(PA9?C)Q
```

Beispiel 2: (PowerShell nur Windows) Überprüft die Instanz, um den Namen des Schlüsselpaars zu ermitteln, das zum Starten der Instanz verwendet wurde, und versucht dann, die entsprechenden Schlüsselpaar-Daten im Konfigurationsspeicher des AWS Toolkit for Visual Studio zu finden. Wenn die Schlüsselpaar-Daten gefunden werden, wird das Passwort entschlüsselt.

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

Ausgabe:

```
mYZ(PA9?C)Q
```

Beispiel 3: Gibt die verschlüsselten Passwortdaten für die Instanz zurück.

```
Get-EC2PasswordData -InstanceId i-12345678
```

Ausgabe:

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- API-Einheiten finden Sie unter [GetPasswordData AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **ImportImage** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ImportImage`.

CLI

AWS CLI

Um eine VM-Image-Datei zu importieren als AMI

Im folgenden `import-image` Beispiel wird die angegebene Datei importiert OVA.

```
aws ec2 import-image \  
  --disk-containers Format=ova,UserBucket="{S3Bucket=my-import-bucket,S3Key=vms/  
  my-server-vm.ova}"
```

Ausgabe:

```
{  
  "ImportTaskId": "import-ami-1234567890abcdef0",  
  "Progress": "2",  
  "SnapshotDetails": [  
    {  
      "DiskImageSize": 0.0,  
      "Format": "ova",
```

```

        "UserBucket": {
            "S3Bucket": "my-import-bucket",
            "S3Key": "vms/my-server-vm.ova"
        }
    },
    "Status": "active",
    "StatusMessage": "pending"
}

```

- API-Einheiten finden Sie [ImportImage](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein Image einer virtuellen Maschine mit einer Festplatte aus dem angegebenen Amazon S3 S3-Bucket EC2 mit einem Idempotenz-Token nach Amazon importiert. Das Beispiel erfordert, dass eine VM-Import-Servicerolle mit dem Standardnamen „vimport“ vorhanden ist, mit einer Richtlinie, die Amazon EC2 Zugriff auf den angegebenen Bucket gewährt, wie im Thema VM-Importvoraussetzungen erklärt. Um eine benutzerdefinierte Rolle zu verwenden, geben Sie den Rollennamen mithilfe des Parameters an. **-RoleName**

```

$container = New-Object Amazon.EC2.Model.ImageDiskContainer
$container.Format="VMDK"
$container.UserBucket = New-Object Amazon.EC2.Model.UserBucket
$container.UserBucket.S3Bucket = "amzn-s3-demo-bucket"
$container.UserBucket.S3Key = "Win_2008_Server_Standard_SP2_64-bit-disk1.vmdk"

$params = @{
    "ClientToken"="idempotencyToken"
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
    "LicenseType"="AWS"
}

Import-EC2Image -DiskContainer $container @params

```

Ausgabe:

```
Architecture      :
```

```
Description      : Windows 2008 Standard Image
Hypervisor       :
ImageId          :
ImportTaskId     : import-ami-abcdefgh
LicenseType     : AWS
Platform        : Windows
Progress        : 2
SnapshotDetails  : {}
Status           : active
StatusMessage    : pending
```

- API-Einheiten finden Sie unter [ImportImage AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **ImportKeyPair** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ImportKeyPair`.

CLI

AWS CLI

Um einen öffentlichen Schlüssel zu importieren

Generieren Sie zunächst ein key pair mit dem Tool Ihrer Wahl. Verwenden Sie zum Beispiel diesen `ssh-keygen`-Befehl:

Befehl:

```
ssh-keygen -t rsa -C "my-key" -f ~/.ssh/my-key
```

Ausgabe:

```
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ec2-user/.ssh/my-key.
Your public key has been saved in /home/ec2-user/.ssh/my-key.pub.
```

...

Dieser Beispielbefehl importiert den angegebenen öffentlichen Schlüssel.

Befehl:

```
aws ec2 import-key-pair --key-name "my-key" --public-key-material fileb://~/.ssh/my-key.pub
```

Ausgabe:

```
{
  "KeyName": "my-key",
  "KeyFingerprint": "1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca"
}
```

- API-Einheiten finden Sie [ImportKeyPair](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein öffentlicher Schlüssel importiert. In der ersten Zeile wird der Inhalt der Datei mit dem öffentlichen Schlüssel (*.pub) in der Variablen **\$publickey** gespeichert. Als Nächstes konvertiert das Beispiel das UTF8 Format der Datei mit dem öffentlichen Schlüssel in eine Base64-kodierte Zeichenfolge und speichert die konvertierte Zeichenfolge in der Variablen **\$pkbase64**. In der letzten Zeile wird der konvertierte öffentliche Schlüssel importiert. EC2 Das Cmdlet gibt den Fingerabdruck und den Namen des Schlüssels als Ergebnisse zurück.

```
$publickey=[Io.File]::ReadAllText("C:\Users\TestUser\.ssh\id_rsa.pub")
$pkbase64 =
[System.Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($publickey))
Import-EC2KeyPair -KeyName Example-user-key -PublicKey $pkbase64
```

Ausgabe:

```
KeyFingerprint      KeyName
-----
-----
```

```
do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1 Example-user-key
```

- API-Einheiten finden Sie unter [ImportKeyPair AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **ImportSnapshot** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ImportSnapshot`.

CLI

AWS CLI

Um einen Snapshot zu importieren

Im folgenden `import-snapshot` Beispiel wird die angegebene Festplatte als Snapshot importiert.

```
aws ec2 import-snapshot \  
  --description "My server VMDK" \  
  --disk-container Format=VMDK,UserBucket={S3Bucket=my-import-bucket,S3Key=vms/  
my-server-vm.vmdk}
```

Ausgabe:

```
{  
  "Description": "My server VMDK",  
  "ImportTaskId": "import-snap-1234567890abcdef0",  
  "SnapshotTaskDetail": {  
    "Description": "My server VMDK",  
    "DiskImageSize": "0.0",  
    "Format": "VMDK",  
    "Progress": "3",  
    "Status": "active",  
    "StatusMessage": "pending"  
  },  
  "UserBucket": {  
    "S3Bucket": "my-import-bucket",  
    "S3Key": "vms/my-server-vm.vmdk"  
  }  
}
```

```

    }
  }
}

```

- API-Einheiten finden Sie [ImportSnapshot](#) unter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird ein VM-Festplatten-Image des Formats 'VMDK' in einen EBS Amazon-Snapshot importiert. Das Beispiel erfordert eine VM-Import-Service-Rolle mit dem Standardnamen „vmimport“ mit einer Richtlinie, die Amazon den EC2-Zugriff auf den angegebenen Bucket ermöglicht, wie im **VM Import Prerequisites** Thema unter <http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/VMImportPrerequisites.html>. Um eine benutzerdefinierte Rolle zu verwenden, geben Sie den Rollennamen mithilfe des **-RoleName** Parameters an.

```

$params = @{
    "ClientToken"="idempotencyToken"
    "Description"="Disk Image Import"
    "DiskContainer_Description" = "Data disk"
    "DiskContainer_Format" = "VMDK"
    "DiskContainer_S3Bucket" = "amzn-s3-demo-bucket"
    "DiskContainer_S3Key" = "datadiskimage.vmdk"
}

Import-EC2Snapshot @params

```

Ausgabe:

Description	ImportTaskId	SnapshotTaskDetail
Disk Image Import	import-snap-abcdefgh	Amazon.EC2.Model.SnapshotTaskDetail

- API-Einheiten finden Sie unter [ImportSnapshot AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **ModifyCapacityReservation** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ModifyCapacityReservation`.

CLI

AWS CLI

Beispiel 1: Um die Anzahl der Instanzen zu ändern, die durch eine bestehende Kapazitätsreservierung reserviert wurden

Im folgenden `modify-capacity-reservation` Beispiel wird die Anzahl der Instances geändert, für die durch die Kapazitätsreservierung Kapazität reserviert wird.

```
aws ec2 modify-capacity-reservation \  
  --capacity-reservation-id cr-1234abcd56EXAMPLE \  
  --instance-count 5
```

Ausgabe:

```
{  
  "Return": true  
}
```

Beispiel 2: Um das Enddatum und die Endzeit für eine bestehende Kapazitätsreservierung zu ändern

Im folgenden `modify-capacity-reservation` Beispiel wird eine bestehende Kapazitätsreservierung so geändert, dass sie am angegebenen Datum und zur angegebenen Uhrzeit endet.

```
aws ec2 modify-capacity-reservation \  
  --capacity-reservation-id cr-1234abcd56EXAMPLE \  
  --end-date-type Limited \  
  --end-date 2019-08-31T23:59:59Z
```


Weitere Informationen finden Sie unter [Ändern einer Kapazitätsreservierung](#) im Amazon Elastic Compute Cloud-Benutzerhandbuch für Linux-Instances.

- API-Einheiten finden Sie [ModifyCapacityReservation](#) unter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird CapacityReservationId cr-0c1f2345db6f7cdba geändert, indem die Anzahl der Instanzen auf 1 geändert wird

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -
InstanceCount 1
```

Ausgabe:

```
True
```

- API-Einheiten [ModifyCapacityReservation AWS Tools for PowerShell](#) finden Sie unter Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **ModifyHosts** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ModifyHosts`.

CLI

AWS CLI

Beispiel 1: So aktivieren Sie die automatische Platzierung für einen Dedicated Host

Das folgende `modify-hosts` Beispiel aktiviert die automatische Platzierung für einen Dedicated Host, sodass er alle Instance-Starts ohne Targeting akzeptiert, die seiner Instance-Typ-Konfiguration entsprechen.

```
aws ec2 modify-hosts \
```

```
--host-id h-06c2f189b4EXAMPLE \  
--auto-placement on
```

Ausgabe:

```
{  
  "Successful": [  
    "h-06c2f189b4EXAMPLE"  
  ],  
  "Unsuccessful": []  
}
```

Beispiel 2: Um die Host-Wiederherstellung für einen Dedicated Host zu aktivieren

Das folgende `modify-hosts` Beispiel aktiviert die Host-Wiederherstellung für den angegebenen Dedicated Host.

```
aws ec2 modify-hosts \  
  --host-id h-06c2f189b4EXAMPLE \  
  --host-recovery on
```

Ausgabe:

```
{  
  "Successful": [  
    "h-06c2f189b4EXAMPLE"  
  ],  
  "Unsuccessful": []  
}
```

Weitere Informationen finden Sie unter [Modifying Dedicated Host Auto Placement](#) im Amazon Elastic Compute Cloud-Benutzerhandbuch für Linux-Instances.

- API-Einheiten finden Sie [ModifyHosts](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die AutoPlacement-Einstellungen für den dedizierten Host `h-01e23f4cd567890f3` auf `Aus` geändert

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

Ausgabe:

```
Successful          Unsuccessful
-----
{h-01e23f4cd567890f3} {}
```

- API-Einheiten [ModifyHosts AWS Tools for PowerShell](#) finden Sie unter Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **ModifyIdFormat** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ModifyIdFormat`.

CLI

AWS CLI

Um das längere ID-Format für eine Ressource zu aktivieren

Im folgenden `modify-id-format` Beispiel wird das längere ID-Format für den `instance` Ressourcentyp aktiviert.

```
aws ec2 modify-id-format \
  --resource instance \
  --use-long-ids
```

Um das längere ID-Format für eine Ressource zu deaktivieren

Im folgenden `modify-id-format` Beispiel wird das längere ID-Format für den `instance` Ressourcentyp deaktiviert.

```
aws ec2 modify-id-format \
  --resource instance \
  --no-use-long-ids
```

Im folgenden `modify-id-format` Beispiel wird das längere ID-Format für alle unterstützten Ressourcentypen aktiviert, die sich innerhalb ihres Anmeldezeitraums befinden.

```
aws ec2 modify-id-format \  
  --resource all-current \  
  --use-long-ids
```

- API-Einheiten finden Sie unter [ModifyIdFormat AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das längere ID-Format für den angegebenen Ressourcentyp aktiviert.

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

Beispiel 2: In diesem Beispiel wird das längere ID-Format für den angegebenen Ressourcentyp deaktiviert.

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- API-Einheiten finden Sie unter [ModifyIdFormat AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **ModifyImageAttribute** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ModifyImageAttribute`.

CLI

AWS CLI

Beispiel 1: Um ein Dokument zu AMI veröffentlichen

Im folgenden `modify-instance-attribute` Beispiel wird das angegebene Objekt AMI öffentlich gemacht.

```
aws ec2 modify-image-attribute \  
  --image-id ami-5731123e \  
  --launch-permission "Add=[{Group=all}]"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 2: Um eine AMI private zu erstellen

Das folgende `modify-instance-attribute` Beispiel macht das angegebene Objekt AMI privat.

```
aws ec2 modify-image-attribute \  
  --image-id ami-5731123e \  
  --launch-permission "Remove=[{Group=all}]"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 3: Um einem AWS Konto die Startberechtigung zu erteilen

Im folgenden `modify-instance-attribute` Beispiel werden dem angegebenen AWS Konto Startberechtigungen erteilt.

```
aws ec2 modify-image-attribute \  
  --image-id ami-5731123e \  
  --launch-permission "Add=[{UserId=123456789012}]"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 4: Um einem AWS Konto die Startberechtigung zu entziehen

Im folgenden `modify-instance-attribute` Beispiel werden die Startberechtigungen aus dem angegebenen AWS Konto entfernt.

```
aws ec2 modify-image-attribute \  
  --image-id ami-5731123e \  
  --launch-permission "Remove=[{UserId=123456789012}]"
```

- API-Einheiten finden Sie [ModifyImageAttribute](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Beschreibung für das angegebene Objekt aktualisiertAMI.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

Beispiel 2: In diesem Beispiel wird die Datei AMI veröffentlicht (zum Beispiel, damit sie von jedem verwendet AWS-Konto werden kann).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType add -UserGroup all
```

Beispiel 3: In diesem Beispiel wird die Datei AMI privat (zum Beispiel, sodass nur Sie als Eigentümer sie verwenden können).

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType remove -UserGroup all
```

Beispiel 4: In diesem Beispiel wird dem angegebenen Benutzer die Startberechtigung erteilt AWS-Konto.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType add -UserId 111122223333
```

Beispiel 5: In diesem Beispiel wird die Startberechtigung für das angegebene Objekt entfernt AWS-Konto.

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -  
OperationType remove -UserId 111122223333
```

- APIEinzelheiten finden Sie unter [ModifyImageAttribute AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter. [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie `ModifyInstanceAttribute` mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ModifyInstanceAttribute`.

CLI

AWS CLI

Beispiel 1: Um den Instanztyp zu ändern

Im folgenden `modify-instance-attribute` Beispiel wird der Instanztyp der angegebenen Instanz geändert. Die Instance muss sich im Status `stopped` befinden.

```
aws ec2 modify-instance-attribute \  
  --instance-id i-1234567890abcdef0 \  
  --instance-type "{\"Value\": \"m1.small\"}"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 2: Um Enhanced Networking auf einer Instance zu aktivieren

Das folgende `modify-instance-attribute` Beispiel aktiviert Enhanced Networking für die angegebene Instanz. Die Instance muss sich im Status `stopped` befinden.

```
aws ec2 modify-instance-attribute \  
  --instance-id i-1234567890abcdef0 \  
  --sriov-net-support simple
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 3: Um das `sourceDestCheck` Attribut zu ändern

Im folgenden `modify-instance-attribute` Beispiel wird das `sourceDestCheck` Attribut der angegebenen Instanz auf `gesetzttrue`. Die Instanz muss sich in einer befinden VPC.

```
aws ec2 modify-instance-attribute --instance-id i-1234567890abcdef0 --source-dest-check "{\"Value\": true}"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 4: Um das `deleteOnTermination` Attribut des Root-Volumes zu ändern

Im folgenden `modify-instance-attribute` Beispiel wird das `deleteOnTermination` Attribut für das Root-Volume der angegebenen EBS Amazon-gestützten Instance auf `false` festgelegt. Standardmäßig gilt dieses Attribut `true` für das Root-Volume.

Befehl:

```
aws ec2 modify-instance-attribute \  
  --instance-id i-1234567890abcdef0 \  
  --block-device-mappings "[{\"DeviceName\": \"/dev/sda1\"},{\"Ebs\":  
{\"DeleteOnTermination\":false}}]"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 5: Um die an eine Instanz angehängten Benutzerdaten zu ändern

Im folgenden `modify-instance-attribute` Beispiel wird der Inhalt der Datei `UserData.txt` als `UserData` für die angegebene Instanz hinzugefügt.

Inhalt der Originaldatei `UserData.txt`:

```
#!/bin/bash  
yum update -y  
service httpd start  
chkconfig httpd on
```

Der Inhalt der Datei muss Base64-codiert sein. Der erste Befehl konvertiert die Textdatei in Base64 und speichert sie als neue Datei.

Linux/MacOS-Version des Befehls:

```
base64 UserData.txt > UserData.base64.txt
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Windows-Version des Befehls:

```
certutil -encode UserData.txt tmp.b64 && findstr /v /c:- tmp.b64 >  
UserData.base64.txt
```

Ausgabe:

```
Input Length = 67
```



```
Output Length = 152
CertUtil: -encode command completed successfully.
```

Jetzt können Sie im folgenden CLI Befehl auf diese Datei verweisen:

```
aws ec2 modify-instance-attribute \
  --instance-id=i-09b5a14dbca622e76 \
  --attribute userData --value file://UserData.base64.txt
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Benutzerdaten und AWS CLI im EC2 Benutzerhandbuch](#).

- API-Einheiten finden Sie [ModifyInstanceAttribute](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Instanztyp der angegebenen Instanz geändert.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

Beispiel 2: In diesem Beispiel wird Enhanced Networking für die angegebene Instanz aktiviert, indem „simple“ als Wert für den Netzwerkunterstützungsparameter Single Root I/O Virtualization (SR-IOV) angegeben wird, -.. SriovNetSupport

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

Beispiel 3: In diesem Beispiel werden die Sicherheitsgruppen für die angegebene Instanz geändert. Die Instanz muss sich in einer VPC befinden. Sie müssen die ID jeder Sicherheitsgruppe angeben, nicht den Namen.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",
  "sg-45678901" )
```

Beispiel 4: Dieses Beispiel aktiviert die EBS I/O-Optimierung für die angegebene Instanz. Diese Funktion ist nicht für alle Instance-Typen verfügbar. Bei Verwendung einer für EBS - optimierten Instanz fallen zusätzliche Nutzungsgebühren an.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

Beispiel 5: In diesem Beispiel wird die Quell-/Zielüberprüfung für die angegebene Instanz aktiviert. Damit eine NAT Instanz ausgeführt werden kann NAT, muss der Wert „falsch“ sein.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

Beispiel 6: In diesem Beispiel wird die Kündigung für die angegebene Instanz deaktiviert.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

Beispiel 7: In diesem Beispiel wird die angegebene Instanz so geändert, dass sie beendet wird, wenn das Herunterfahren von der Instance aus initiiert wird.

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -  
InstanceInitiatedShutdownBehavior terminate
```

- API-Einheiten finden Sie unter [ModifyInstanceAttribute AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **ModifyInstanceCreditSpecification** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ModifyInstanceCreditSpecification`.

CLI

AWS CLI

Um die Kreditoption für die CPU Nutzung einer Instanz zu ändern

In diesem Beispiel wird die Kreditoption für die CPU Nutzung der angegebenen Instanz in der angegebenen Region auf „unbegrenzt“ geändert. Gültige Kreditoptionen sind „Standard“ und „unbegrenzt“.

Befehl:

```
aws ec2 modify-instance-credit-specification --instance-credit-specification "InstanceId=i-1234567890abcdef0,CpuCredits=unlimited"
```

Ausgabe:

```
{
  "SuccessfulInstanceCreditSpecifications": [
    {
      "InstanceId": "i-1234567890abcdef0"
    }
  ],
  "UnsuccessfulInstanceCreditSpecifications": []
}
```

- API-Einheiten finden Sie [ModifyInstanceCreditSpecification](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dies ermöglicht unbegrenzte T2-Credits, zum Beispiel i-01234567890abcdef.

```
$Credit = New-Object -TypeName
  Amazon.EC2.Model.InstanceCreditSpecificationRequest
$Credit.InstanceId = "i-01234567890abcdef"
$Credit.CpuCredits = "unlimited"
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- API-Einheiten [ModifyInstanceCreditSpecification](#) finden AWS Tools for PowerShell Sie unter Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **ModifyNetworkInterfaceAttribute** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ModifyNetworkInterfaceAttribute`.

CLI

AWS CLI

Um das Attachment-Attribut einer Netzwerkschnittstelle zu ändern

Dieser Beispielbefehl ändert das attachment Attribut der angegebenen Netzwerkschnittstelle.

Befehl:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --  
attachment AttachmentId=eni-attach-43348162,DeleteOnTermination=false
```

Um das Beschreibungsattribut einer Netzwerkschnittstelle zu ändern

Dieser Beispielbefehl ändert das description Attribut der angegebenen Netzwerkschnittstelle.

Befehl:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --  
description "My description"
```

Um das groupSet Attribut einer Netzwerkschnittstelle zu ändern

Dieser Beispielbefehl ändert das groupSet Attribut der angegebenen Netzwerkschnittstelle.

Befehl:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --  
groups sg-903004f8 sg-1a2b3c4d
```

Um das sourceDestCheck Attribut einer Netzwerkschnittstelle zu ändern

Dieser Beispielbefehl ändert das sourceDestCheck Attribut der angegebenen Netzwerkschnittstelle.

Befehl:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --  
no-source-dest-check
```

- API-Einheiten finden Sie unter [ModifyNetworkInterfaceAttribute AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Netzwerkschnittstelle so geändert, dass die angegebene Anlage beim Beenden gelöscht wird.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -  
Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

Beispiel 2: In diesem Beispiel wird die Beschreibung der angegebenen Netzwerkschnittstelle geändert.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description  
"my description"
```

Beispiel 3: In diesem Beispiel wird die Sicherheitsgruppe für die angegebene Netzwerkschnittstelle geändert.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups  
sg-1a2b3c4d
```

Beispiel 4: In diesem Beispiel wird die Quell-/Zielüberprüfung für die angegebene Netzwerkschnittstelle deaktiviert.

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -  
SourceDestCheck $false
```

- API-Einheiten finden Sie unter [ModifyNetworkInterfaceAttribute](#) Cmdlet-Referenz. [AWS Tools for PowerShell](#)

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie `ModifyReservedInstances` mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ModifyReservedInstances`.

CLI

AWS CLI

Um Reserved Instances zu ändern

Mit diesem Beispielbefehl wird eine Reserved Instance in eine andere Availability Zone in derselben Region verschoben.

Befehl:

```
aws ec2 modify-reserved-instances --reserved-instances-ids b847fa93-  
e282-4f55-b59a-1342f5bd7c02 --target-configurations AvailabilityZone=us-  
west-1c,Platform=EC2-Classic,InstanceCount=10
```

Ausgabe:

```
{  
  "ReservedInstancesModificationId": "rimod-d3ed4335-b1d3-4de6-ab31-0f13aaf46687"  
}
```

Um die Netzwerkplattform von Reserved Instances zu ändern

Dieser Beispielbefehl konvertiert EC2 -Classic Reserved Instances in EC2 -VPC.

Befehl:

```
aws ec2 modify-reserved-instances --reserved-instances-ids f127bd27-  
edb7-44c9-a0eb-0d7e09259af0 --target-configurations AvailabilityZone=us-  
west-1c,Platform=EC2-VPC,InstanceCount=5
```

Ausgabe:

```
{  
  "ReservedInstancesModificationId": "rimod-82fa9020-668f-4fb6-945d-61537009d291"  
}
```

Weitere Informationen finden Sie unter [Modifying Your Reserved Instances](#) im EC2Amazon-Benutzerhandbuch.

So ändern Sie die Instance-Größe von Reserved Instances

Mit diesem Beispielbefehl wird eine Reserved Instance mit 10 m1.small UNIX Linux/-Instances in us-west-1c geändert, sodass aus 8 m1.small-Instances 2 m1.large-Instances werden und die verbleibenden 2 m1.small zu 1 m1.medium-Instance in derselben Availability Zone werden.
Befehl:

```
aws ec2 modify-reserved-instances --reserved-instances-ids 1ba8e2e3-3556-4264-949e-63ee671405a9 --target-configurations AvailabilityZone=us-west-1c,Platform=EC2-Classic,InstanceCount=2,InstanceType=m1.large AvailabilityZone=us-west-1c,Platform=EC2-Classic,InstanceCount=1,InstanceType=m1.medium
```

Ausgabe:

```
{
  "ReservedInstancesModificationId": "rimod-acc5f240-080d-4717-b3e3-1c6b11fa00b6"
}
```

Weitere Informationen finden Sie unter [Ändern der Instance-Größe Ihrer Reservierungen](#) im EC2Amazon-Benutzerhandbuch.

- APIEinzelheiten finden Sie [ModifyReservedInstances](#) unter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel werden die Availability Zone, die Anzahl der Instanzen und die Plattform für die angegebenen Reserved Instances geändert.

```
$config = New-Object Amazon.EC2.Model.ReservedInstancesConfiguration
$config.AvailabilityZone = "us-west-2a"
$config.InstanceCount = 1
$config.Platform = "EC2-VPCC"

Edit-EC2ReservedInstance `
```

```
-ReservedInstancesId @"FE32132D-70D5-4795-B400-AE435EXAMPLE",
"0CC556F3-7AB8-4C00-B0E5-98666EXAMPLE") `
-TargetConfiguration $config
```

- API-Einheiten finden Sie unter [ModifyReservedInstances AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **ModifySnapshotAttribute** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ModifySnapshotAttribute`.

CLI

AWS CLI

Beispiel 1: Um ein Snapshot-Attribut zu ändern

Im folgenden `modify-snapshot-attribute` Beispiel wird das `createVolumePermission` Attribut für den angegebenen Snapshot aktualisiert, wodurch die Volumenberechtigungen für den angegebenen Benutzer entfernt werden.

```
aws ec2 modify-snapshot-attribute \
  --snapshot-id snap-1234567890abcdef0 \
  --attribute createVolumePermission \
  --operation-type remove \
  --user-ids 123456789012
```

Beispiel 2: Um einen Snapshot öffentlich zu machen

Das folgende `modify-snapshot-attribute` Beispiel macht den angegebenen Snapshot öffentlich.

```
aws ec2 modify-snapshot-attribute \
  --snapshot-id snap-1234567890abcdef0 \
  --attribute createVolumePermission \
  --operation-type add \
  --group-names all
```


- API-Einheiten finden Sie [ModifySnapshotAttribute](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der angegebene Snapshot veröffentlicht, indem es sein `CreateVolumePermission` Attribut festlegt.

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute  
CreateVolumePermission -OperationType Add -GroupName all
```

- API-Einheiten finden Sie unter [ModifySnapshotAttribute AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **ModifySpotFleetRequest** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ModifySpotFleetRequest`.

CLI

AWS CLI

Um eine Spot-Flottenanfrage zu ändern

Dieser Beispielbefehl aktualisiert die Zielkapazität der angegebenen Spot-Flottenanforderung.

Befehl:

```
aws ec2 modify-spot-fleet-request --target-capacity 20 --spot-fleet-request-  
id sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

Ausgabe:

```
{  
  "Return": true
```

```
}
```

Dieser Beispielbefehl verringert die Zielkapazität der angegebenen Spot-Flottenanforderung, ohne dass dadurch Spot-Instances beendet werden.

Befehl:

```
aws ec2 modify-spot-fleet-request --target-capacity 10 --excess-capacity-termination-policy NoTermination --spot-fleet-request-ids sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

Ausgabe:

```
{
  "Return": true
}
```

- API-Einheiten finden Sie [ModifySpotFleetRequest](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Zielkapazität der angegebenen Spot-Flottenanfrage aktualisiert.

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

Ausgabe:

```
True
```

- API-Einheiten finden Sie unter [ModifySpotFleetRequest AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **ModifySubnetAttribute** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt **ModifySubnetAttribute**.

CLI

AWS CLI

Um das Verhalten eines Subnetzes bei der öffentlichen IPv4 Adressierung zu ändern

In diesem Beispiel wird das Subnetz-1a2b3c4d dahingehend geändert, dass allen in diesem Subnetz gestarteten Instances eine öffentliche Adresse zugewiesen wird. IPv4 Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 modify-subnet-attribute --subnet-id subnet-1a2b3c4d --map-public-ip-on-launch
```

Um IPv6 das Adressierungsverhalten eines Subnetzes zu ändern

In diesem Beispiel wird Subnetz-1a2b3c4d dahingehend geändert, dass allen in diesem Subnetz gestarteten Instances eine Adresse aus dem Bereich des Subnetzes zugewiesen wird. IPv6

Befehl:

```
aws ec2 modify-subnet-attribute --subnet-id subnet-1a2b3c4d --assign-ipv6-address-on-creation
```

Weitere Informationen finden Sie unter IP-Adressierung in Ihrem im Virtual Private Cloud-Benutzerhandbuch. VPC AWS

- API-Einheiten finden Sie [ModifySubnetAttribute](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die öffentliche IP-Adressierung für das angegebene Subnetz aktiviert.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

Beispiel 2: In diesem Beispiel wird die öffentliche IP-Adressierung für das angegebene Subnetz deaktiviert.

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- API-Einheiten finden Sie unter [ModifySubnetAttribute AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **ModifyVolumeAttribute** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ModifyVolumeAttribute`.

CLI

AWS CLI

Um ein Volumenattribut zu ändern

In diesem Beispiel wird das `autoEnableIo` Attribut des Volumes mit der ID `vol-1234567890abcdef0` auf `true` festgelegt. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 modify-volume-attribute --volume-id vol-1234567890abcdef0 --auto-enable-io
```

- API-Einheiten finden Sie [ModifyVolumeAttribute](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Attribut des angegebenen Volumes geändert. I/O-Operationen für das Volume werden automatisch wieder aufgenommen, nachdem sie aufgrund potenziell inkonsistenter Daten unterbrochen wurden.

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- API-Einheiten finden Sie unter [ModifyVolumeAttribute AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **ModifyVpcAttribute** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ModifyVpcAttribute`.

CLI

AWS CLI

Um das `enableDnsSupport` Attribut zu ändern

In diesem Beispiel wird das `enableDnsSupport` Attribut geändert. Dieses Attribut gibt an, ob die DNS-Auflösung für aktiviert ist. VPC. Wenn dieses Attribut zutrifft `true`, löst der DNS-Amazon-Server DNS-Hostnamen für Ihre Instanzen in die entsprechenden IP-Adressen auf; andernfalls nicht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 modify-vpc-attribute --vpc-id vpc-a01106c2 --enable-dns-support "{\"Value\":false}"
```

Um das Attribut zu ändern `enableDnsHostnames`

In diesem Beispiel wird das `enableDnsHostnames` Attribut geändert. Dieses Attribut gibt an, ob Instanzen im VPC Get DNS Hostnames gestartet wurden. Wenn dieses Attribut

zutritt `true`, VPC erhalten Instanzen in der DNS Gruppe Hostnamen, andernfalls nicht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 modify-vpc-attribute --vpc-id vpc-a01106c2 --enable-dns-hostnames  
"{\"Value\":false}"
```

- API Einzelheiten finden Sie [ModifyVpcAttribute](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel aktiviert die Unterstützung von DNS Hostnamen für die angegebenen VPC.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

Beispiel 2: In diesem Beispiel wird die Unterstützung für DNS Hostnamen für die angegebenen Daten deaktiviert. VPC

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

Beispiel 3: In diesem Beispiel wird die Unterstützung für die DNS Auflösung der angegebenen Daten aktiviert. VPC

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

Beispiel 4: In diesem Beispiel wird die Unterstützung für die DNS Auflösung für die angegebene VPC Datei deaktiviert.

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- API Einzelheiten finden Sie unter [ModifyVpcAttribute AWS Tools for PowerShell Cmdlet](#)-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **MonitorInstances** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `MonitorInstances`.

C++

SDK für C++

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
//! Enable detailed monitoring for an Amazon Elastic Compute Cloud (Amazon EC2)
instance.
/*!
 \param instanceId: An EC2 instance ID.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::EC2::enableMonitoring(const Aws::String &instanceId,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::MonitorInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::MonitorInstancesOutcome dryRunOutcome =
ec2Client.MonitorInstances(request);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to enable monitoring on instance. A dry run
should trigger an error."
            <<
            std::endl;
        return false;
    }
}
```

```

    } else if (dryRunOutcome.GetError().GetErrorType()
               != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cerr << "Failed dry run to enable monitoring on instance " <<
                  instanceId << ": " << dryRunOutcome.GetError().GetMessage() <<
                  std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::MonitorInstancesOutcome monitorInstancesOutcome =
ec2Client.MonitorInstances(request);
    if (!monitorInstancesOutcome.IsSuccess()) {
        std::cerr << "Failed to enable monitoring on instance " <<
                  instanceId << ": " <<
                  monitorInstancesOutcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully enabled monitoring on instance " <<
                  instanceId << std::endl;
    }

    return monitorInstancesOutcome.IsSuccess();
}

```

- API-Einheiten finden Sie [MonitorInstances](#) in der AWS SDK for C++ API-Referenz.

CLI

AWS CLI

So aktivieren Sie eine detaillierte Überwachung für eine Instance

Dieser Beispielbefehl aktiviert die detaillierte Überwachung für die angegebene Instance.

Befehl:

```
aws ec2 monitor-instances --instance-ids i-1234567890abcdef0
```

Ausgabe:

```
{
  "InstanceMonitorings": [
    {
```



```
    "InstanceId": "i-1234567890abcdef0",
    "Monitoring": {
      "State": "pending"
    }
  ]
}
```

- API-Einheiten finden Sie [MonitorInstances](#) unter AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import { EC2Client, MonitorInstancesCommand } from "@aws-sdk/client-ec2";

/**
 * Turn on detailed monitoring for the selected instance.
 * By default, metrics are sent to Amazon CloudWatch every 5 minutes.
 * For a cost you can enable detailed monitoring which sends metrics every
 * minute.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new MonitorInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { InstanceMonitorings } = await client.send(command);
    const instancesBeingMonitored = InstanceMonitorings.map(
      (im) =>
        ` • Detailed monitoring state for ${im.InstanceId} is
        ${im.Monitoring.State}.`,
    );
  }
};
```

```

    console.log("Monitoring status:");
    console.log(instancesBeingMonitored.join("\n"));
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};

```

- API-Einheiten finden Sie [MonitorInstances](#) in der AWS SDK for JavaScript API-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel ermöglicht eine detaillierte Überwachung für die angegebene Instanz.

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

Ausgabe:

InstanceId	Monitoring
-----	-----
i-12345678	Amazon.EC2.Model.Monitoring

- API-Einheiten finden Sie unter [MonitorInstances AWS Tools for PowerShell Cmdlet-Referenz](#).

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
  APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
  lt_instance_ids.

  "Perform dry run"
  TRY.
    " DryRun is set to true. This checks for the required permissions to
    monitor the instance without actually making the request. "
    lo_ec2->monitorinstances(
      it_instanceids = lt_instance_ids
      iv_dryrun = abap_true
    ).
  CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    " If the error code returned is `DryRunOperation`, then you have the
    required permissions to monitor this instance. "
    IF lo_exception->av_err_code = 'DryRunOperation'.
      MESSAGE 'Dry run to enable detailed monitoring completed.' TYPE 'I'.
      " DryRun is set to false to enable detailed monitoring. "
      lo_ec2->monitorinstances(
        it_instanceids = lt_instance_ids
        iv_dryrun = abap_false
      ).
      MESSAGE 'Detailed monitoring enabled.' TYPE 'I'.
    " If the error code returned is `UnauthorizedOperation`, then you don't
    have the required permissions to monitor this instance. "
    ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
      MESSAGE 'Dry run to enable detailed monitoring failed. User does not
      have the permissions to monitor the instance.' TYPE 'E'.
    ELSE.
      DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
      >av_err_msg }|.
      MESSAGE lv_error TYPE 'E'.
    ENDIF.
  ENDMETHOD.
ENDCLASS.

```

- APIEinzelheiten finden Sie [MonitorInstances](#) unter AWS SDKSAPABAPAPI als Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **MoveAddressToVpc** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `MoveAddressToVpc`.

CLI

AWS CLI

Um eine Adresse nach EC2 - zu verschieben VPC

In diesem Beispiel wird die Elastic IP-Adresse 54.123.4.56 auf die Plattform - verschoben. EC2 VPC

Befehl:

```
aws ec2 move-address-to-vpc --public-ip 54.123.4.56
```

Ausgabe:

```
{
  "Status": "MoveInProgress"
}
```

- API Einzelheiten finden Sie in der Befehlsreferenz [MoveAddressToVpc](#). AWS CLI

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine EC2 Instance mit der öffentlichen IP-Adresse 12.345.67.89 auf die VPC Plattform EC2 - in der Region USA Ost (Nord-Virginia) verschoben.

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

Beispiel 2: In diesem Beispiel werden die Ergebnisse eines Befehls über die Pipeline an das Cmdlet übergeben `Get-EC2Instance`. `Move-EC2AddressToVpc` Der `Get-EC2Instance` Befehl

ruft eine Instanz ab, die durch die Instanz-ID angegeben ist, und gibt dann die öffentliche IP-Adresseigenschaft der Instanz zurück.

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-EC2AddressToVpc
```

- API-Einheiten finden Sie unter [MoveAddressToVpc AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PurchaseHostReservation** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `PurchaseHostReservation`.

CLI

AWS CLI

Um eine Reservierung für einen Dedicated Host zu erwerben

In diesem Beispiel wird das angegebene Reservierungsangebot für Dedicated Hosts für den angegebenen Dedicated Host in Ihrem Konto erworben.

Befehl:

```
aws ec2 purchase-host-reservation --offering-id hro-03f707bf363b6b324 --host-id-set h-013abcd2a00cbd123
```

Ausgabe:

```
{
  "TotalHourlyPrice": "1.499",
  "Purchase": [
    {
      "HourlyPrice": "1.499",
      "InstanceFamily": "m4",
      "PaymentOption": "NoUpfront",
      "HostIdSet": [
        "h-013abcd2a00cbd123"
      ]
    }
  ]
}
```

```

    ],
    "HostReservationId": "hr-0d418a3a4ffc669ae",
    "UpfrontPrice": "0.000",
    "Duration": 31536000
  }
],
"TotalUpfrontPrice": "0.000"
}

```

- API-Einheiten finden Sie [PurchaseHostReservation](#) unter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Reservierungsangebot hro-0c1f23456789d0ab mit Konfigurationen erworben, die denen Ihres Dedicated Hosts h-01e23f4cd567890f1 entsprechen

```

New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet
h-01e23f4cd567890f1

```

Ausgabe:

```

ClientToken      :
CurrencyCode     :
Purchase         : {hr-0123f4b5d67bedc89}
TotalHourlyPrice : 1.307
TotalUpfrontPrice : 0.000

```

- Einzelheiten [PurchaseHostReservation AWS Tools for PowerShell](#) finden Sie unter Cmdlet-Referenz. API

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **PurchaseScheduledInstances** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `PurchaseScheduledInstances`.

CLI

AWS CLI

Um eine geplante Instanz zu erwerben

In diesem Beispiel wird eine Scheduled Instance gekauft.

Befehl:

```
aws ec2 purchase-scheduled-instances --purchase-requests file://purchase-request.json
```

purchase-request.json:

```
[
  {
    "PurchaseToken": "eyJ2IjoiMSIsInMiOjEsImMiOi...",
    "InstanceCount": 1
  }
]
```

Ausgabe:

```
{
  "ScheduledInstanceSet": [
    {
      "AvailabilityZone": "us-west-2b",
      "ScheduledInstanceId": "sci-1234-1234-1234-1234-123456789012",
      "HourlyPrice": "0.095",
      "CreateDate": "2016-01-25T21:43:38.612Z",
      "Recurrence": {
        "OccurrenceDaySet": [
          1
        ],
        "Interval": 1,
        "Frequency": "Weekly",
        "OccurrenceRelativeToEnd": false,
        "OccurrenceUnit": ""
      },
      "Platform": "Linux/UNIX",
      "TermEndDate": "2017-01-31T09:00:00Z",
    }
  ]
}
```

```

        "InstanceCount": 1,
        "SlotDurationInHours": 32,
        "TermStartDate": "2016-01-31T09:00:00Z",
        "NetworkPlatform": "EC2-VPC",
        "TotalScheduledInstanceHours": 1696,
        "NextSlotStartTime": "2016-01-31T09:00:00Z",
        "InstanceType": "c4.large"
    }
]
}

```

- API-Einheiten finden Sie in der Befehlsreferenz [PurchaseScheduledInstances](#). AWS CLI

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine geplante Instance gekauft.

```

$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOjEsImMiOi..."
New-EC2ScheduledInstancePurchase -PurchaseRequest $request

```

Ausgabe:

```

AvailabilityZone      : us-west-2b
CreateDate            : 1/25/2016 1:43:38 PM
HourlyPrice           : 0.095
InstanceCount        : 1
InstanceType         : c4.large
NetworkPlatform      : EC2-VPC
NextSlotStartTime     : 1/31/2016 1:00:00 AM
Platform             : Linux/UNIX
PreviousSlotEndTime   :
Recurrence            : Amazon.EC2.Model.ScheduledInstanceRecurrence
ScheduledInstanceId   : sci-1234-1234-1234-1234-123456789012
SlotDurationInHours  : 32
TermEndDate          : 1/31/2017 1:00:00 AM
TermStartDate        : 1/31/2016 1:00:00 AM
TotalScheduledInstanceHours : 1696

```


- API-Einheiten finden Sie unter [PurchaseScheduledInstances AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter. [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **RebootInstances** mit einem AWS SDK oder CLI


Die folgenden Codebeispiele zeigen, wie man es benutzt `RebootInstances`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erstellen und Verwalten eines ausfallsicheren Services](#)

.NET

AWS SDK for .NET

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Starten Sie eine Instanz anhand ihrer ID neu.

```
/// <summary>
/// Reboot a specific EC2 instance.
/// </summary>
/// <param name="ec2InstanceId">The instance Id of the instance that will be
rebooted.</param>
/// <returns>Async Task.</returns>
public async Task<bool> RebootInstances(string ec2InstanceId)
{
    try
    {
        var request = new RebootInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId },
```

```
};

await _amazonEC2.RebootInstancesAsync(request);

// Wait for the instance to be running.
Console.WriteLine("Waiting for the instance to start.");
await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);

return true;
}
catch (AmazonEC2Exception ec2Exception)
{
    if (ec2Exception.ErrorCode == "InvalidInstanceId")
    {
        _logger.LogError(
            $"InstanceId {ec2InstanceId} is invalid, unable to reboot.
{ec2Exception.Message}");
    }
    return false;
}
catch (Exception ex)
{
    _logger.LogError(
        $"An error occurred while rebooting the instance
{ec2InstanceId}.: {ex.Message}");
    return false;
}
}

/// <summary>
/// Wait until an EC2 instance is in a specified state.
/// </summary>
/// <param name="instanceId">The instance Id.</param>
/// <param name="stateName">The state to wait for.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
{
    var request = new DescribeInstancesRequest
    {
        InstanceIds = new List<string> { instanceId }
    };

    // Wait until the instance is in the specified state.
```

```

    var hasState = false;
    do
    {
        // Wait 5 seconds.
        Thread.Sleep(5000);

        // Check for the desired state.
        var response = await _amazonEC2.DescribeInstancesAsync(request);
        var instance = response.Reservations[0].Instances[0];
        hasState = instance.State.Name == stateName;
        Console.WriteLine(". ");
    } while (!hasState);

    return hasState;
}

```

Ersetzen Sie das Profil für eine Instance, starten Sie einen Webserver neu und starten Sie ihn neu.

```

/// <summary>
/// Replace the profile associated with a running instance. After the profile
is replaced, the instance
/// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
/// used to restart the Python web server.
/// </summary>
/// <param name="instanceId">The Id of the instance to update.</param>
/// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
/// <param name="associationId">The Id of the existing profile association
for the instance.</param>
/// <returns>Async task.</returns>
public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
{
    try
    {
        await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
            new ReplaceIamInstanceProfileAssociationRequest()
            {
                AssociationId = associationId,

```

```
        IamInstanceProfile = new IamInstanceProfileSpecification()
        {
            Name = credsProfileName
        }
    });
// Allow time before resetting.
Thread.Sleep(25000);

await _amazonEc2.RebootInstancesAsync(
    new RebootInstancesRequest(new List<string>() { instanceId }));
Thread.Sleep(25000);
var instanceReady = false;
var retries = 5;
while (retries-- > 0 && !instanceReady)
{
    var instancesPaginator =
        _amazonSsm.Paginators.DescribeInstanceInformation(
            new DescribeInstanceInformationRequest());
    // Get the entire list using the paginator.
    await foreach (var instance in
instancesPaginator.InstanceInformationList)
    {
        instanceReady = instance.InstanceId == instanceId;
        if (instanceReady)
        {
            break;
        }
    }
}
Console.WriteLine("Waiting for instance to be running.");
await WaitForInstanceState(instanceId, InstanceStateName.Running);
Console.WriteLine("Instance ready.");
Console.WriteLine($"Sending restart command to instance
{instanceId}");
await _amazonSsm.SendCommandAsync(
    new SendCommandRequest()
    {
        InstanceIds = new List<string>() { instanceId },
        DocumentName = "AWS-RunShellScript",
        Parameters = new Dictionary<string, List<string>>()
        {
            {
                "commands",
```

```

        new List<string>() { "cd / && sudo python3 server.py
80" }
    }
}
});
    Console.WriteLine($"Restarted the web server on instance
{instanceId}");
}
catch (AmazonEC2Exception ec2Exception)
{
    if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
    {
        _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
    }

    throw;
}
catch (Exception ex)
{
    _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
    throw;
}
}
}

```

- API-Einheiten finden Sie [RebootInstances](#) unter AWS SDK for .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

/*! Reboot an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
    \param instanceID: An EC2 instance ID.

```

```

\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::EC2::rebootInstance(const Aws::String &instanceId,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::RebootInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::RebootInstancesOutcome dry_run_outcome =
ec2Client.RebootInstances(request);
    if (dry_run_outcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to reboot on instance. A dry run should
trigger an error."
            <<
            std::endl;
        return false;
    } else if (dry_run_outcome.GetError().GetErrorType()
                != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to reboot instance " << instanceId << ": "
            << dry_run_outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::RebootInstancesOutcome outcome =
ec2Client.RebootInstances(request);
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to reboot instance " << instanceId << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully rebooted instance " << instanceId <<
            std::endl;
    }

    return outcome.IsSuccess();
}

```

- API-Einheiten finden Sie [RebootInstances](#) in der AWS SDK for C++ API-Referenz.

CLI

AWS CLI

Um eine EC2 Amazon-Instance neu zu starten

In diesem Beispiel wird die angegebene Instance neu gestartet. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 reboot-instances --instance-ids i-1234567890abcdef5
```

Weitere Informationen finden Sie unter Ihre Instance neu starten im Benutzerhandbuch für Amazon Elastic Compute Cloud.

- API-Einheiten finden Sie [RebootInstances](#) in der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import { EC2Client, RebootInstancesCommand } from "@aws-sdk/client-ec2";

/**
 * Requests a reboot of the specified instances. This operation is asynchronous;
 * it only queues a request to reboot the specified instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new RebootInstancesCommand({
    InstanceIds: instanceIds,
  });
```

```
try {
  await client.send(command);
  console.log("Instance rebooted successfully.");
} catch (caught) {
  if (
    caught instanceof Error &&
    caught.name === "InvalidInstanceID.NotFound"
  ) {
    console.warn(
      `${caught.message}. Please provide the InstanceId of a valid instance to
reboot.` ,
    );
  } else {
    throw caught;
  }
}
};
```

- API-Einheiten finden Sie [RebootInstances](#) in der AWS SDK for JavaScript API-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instanz neu gestartet.

```
Restart-EC2Instance -InstanceId i-12345678
```

- API-Einheiten finden Sie unter [RebootInstances AWS Tools for PowerShell](#) Cmdlet-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.


```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
```

```
self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"

# Failure mode
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

def replace_instance_profile(
    self,
    instance_id: str,
    new_instance_profile_name: str,
    profile_association_id: str,
) -> None:
    """
    Replaces the profile associated with a running instance. After the
    profile is
    replaced, the instance is rebooted to ensure that it uses the new
    profile. When
    the instance is ready, Systems Manager is used to restart the Python web
    server.

    :param instance_id: The ID of the instance to restart.
    :param new_instance_profile_name: The name of the new profile to
    associate with
        the specified instance.
    :param profile_association_id: The ID of the existing profile association
    for the
        instance.
    """
    try:
        self.ec2_client.replace_iam_instance_profile_association(
            IamInstanceProfile={"Name": new_instance_profile_name},
            AssociationId=profile_association_id,
        )
        log.info(
            "Replaced instance profile for association %s with profile %s.",
            profile_association_id,
            new_instance_profile_name,
        )
        time.sleep(5)

        self.ec2_client.reboot_instances(InstanceIds=[instance_id])
```

```

log.info("Rebooting instance %s.", instance_id)
waiter = self.ec2_client.get_waiter("instance_running")
log.info("Waiting for instance %s to be running.", instance_id)
waiter.wait(InstanceIds=[instance_id])
log.info("Instance %s is now running.", instance_id)

self.ssm_client.send_command(
    InstanceIds=[instance_id],
    DocumentName="AWS-RunShellScript",
    Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
)
log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
except ClientError as err:
    log.error("Failed to replace instance profile.")
    error_code = err.response["Error"]["Code"]
    if error_code == "InvalidAssociationID.NotFound":
        log.error(
            f"Association ID '{profile_association_id}' does not exist."
            "Please check the association ID and try again."
        )
    if error_code == "InvalidInstanceId":
        log.error(
            f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
            f"Please verify the instance ID and try again."
        )
    log.error(f"Full error:\n\t{err}")

```

- API-Einheiten finden Sie unter [RebootInstances](#) Python (Boto3) API -Referenz.AWS SDK

Rust

SDK für Rust

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
pub async fn reboot(&self, ec2: &EC2) -> Result<(), EC2Error> {
    if self.instance.is_some() {
        ec2.reboot_instance(self.instance_id()).await?;
        ec2.wait_for_instance_stopped(self.instance_id(), None)
            .await?;
        ec2.wait_for_instance_ready(self.instance_id(), None)
            .await?;
    }
    Ok(())
}
```

```
pub async fn reboot_instance(&self, instance_id: &str) -> Result<(),
EC2Error> {
    tracing::info!("Rebooting instance {instance_id}");

    self.client
        .reboot_instances()
        .instance_ids(instance_id)
        .send()
        .await?;

    Ok(())
}
```

Kellner zum Beispiel, um im angehaltenen Zustand zu sein und die Kellner API zu benutzen. Die Verwendung von Waiters API erfordert `use aws_sdk_ec2: :client: :Waiters` in der Rust-Datei.

```
/// Wait for an instance to be ready and status ok (default wait 60 seconds)
pub async fn wait_for_instance_ready(
    &self,
    instance_id: &str,
    duration: Option<Duration>,
) -> Result<(), EC2Error> {
    self.client
        .wait_until_instance_status_ok()
        .instance_ids(instance_id)
        .wait(duration.unwrap_or(Duration::from_secs(60)))
        .await
        .map_err(|err| match err {
```

```

        WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
            "Exceeded max time ({}s) waiting for instance to start.",
            exceeded.max_wait().as_secs()
        )),
        _ => EC2Error::from(err),
    }}?;
    Ok(())
}

pub async fn wait_for_instance_stopped(
    &self,
    instance_id: &str,
    duration: Option<Duration>,
) -> Result<(), EC2Error> {
    self.client
        .wait_until_instance_stopped()
        .instance_ids(instance_id)
        .wait(duration.unwrap_or(Duration::from_secs(60)))
        .await
        .map_err(|err| match err {
            WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                "Exceeded max time ({}s) waiting for instance to stop.",
                exceeded.max_wait().as_secs(),
            )),
            _ => EC2Error::from(err),
        }}?;
    Ok(())
}

```

- Einzelheiten [RebootInstances](#) finden AWS SDK Sie in der Rust-Referenz. API API

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
  APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
  lt_instance_ids.

"Perform dry run"
TRY.
  " DryRun is set to true. This checks for the required permissions to
  reboot the instance without actually making the request. "
  lo_ec2->rebootinstances(
    it_instanceids = lt_instance_ids
    iv_dryrun = abap_true
  ).
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
  " If the error code returned is `DryRunOperation`, then you have the
  required permissions to reboot this instance. "
  IF lo_exception->av_err_code = 'DryRunOperation'.
    MESSAGE 'Dry run to reboot instance completed.' TYPE 'I'.
    " DryRun is set to false to make a reboot request. "
    lo_ec2->rebootinstances(
      it_instanceids = lt_instance_ids
      iv_dryrun = abap_false
    ).
    MESSAGE 'Instance rebooted.' TYPE 'I'.
  " If the error code returned is `UnauthorizedOperation`, then you don't
  have the required permissions to reboot this instance. "
  ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
    MESSAGE 'Dry run to reboot instance failed. User does not have
  permissions to reboot the instance.' TYPE 'E'.
  ELSE.
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
  >av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
  ENDIF.
ENDTRY.
ENDTRY.

```

- API-Einheiten finden Sie [RebootInstances](#) unter AWS SDKSAPABAPI als Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie `RegisterImage` mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `RegisterImage`.

CLI

AWS CLI

Beispiel 1: Um einen AMI mithilfe einer Manifestdatei zu registrieren

Das folgende `register-image` Beispiel registriert eine AMI unter Verwendung der angegebenen Manifestdatei in Amazon S3.

```
aws ec2 register-image \  
  --name my-image \  
  --image-location my-s3-bucket/myimage/image.manifest.xml
```

Ausgabe:

```
{  
  "ImageId": "ami-1234567890EXAMPLE"  
}
```

Weitere Informationen finden Sie unter [Amazon Machine Images \(AMI\)](#) im EC2 Amazon-Benutzerhandbuch.

Beispiel 2: Um einen AMI mithilfe eines Snapshots eines Root-Geräts zu registrieren

Im folgenden `register-image` Beispiel wird ein Benutzer registriert, der den AMI angegebenen Snapshot eines EBS Root-Volumes als Gerät verwendet `/dev/xvda`. Die Blockgerätezuordnung beinhaltet auch ein leeres EBS 100-GiB-Volume als Gerät `/dev/xvdf`.

```
aws ec2 register-image \  
  --name my-image \  
  --root-device-name /dev/xvda \  
  --block-device-mappings DeviceName=/dev/  
xvda,Ebs={SnapshotId=snap-0db2cf683925d191f} DeviceName=/dev/  
xvdf,Ebs={VolumeSize=100}
```

Ausgabe:

```
{
```

```
"ImageId": "ami-1a2b3c4d5eEXAMPLE"  
}
```

Weitere Informationen finden Sie unter [Amazon Machine Images \(AMI\)](#) im EC2Amazon-Benutzerhandbuch.

- APIEinzelheiten finden Sie [RegisterImage](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel registriert eine AMI unter Verwendung der angegebenen Manifestdatei in Amazon S3.

```
Register-EC2Image -ImageLocation amzn-s3-demo-bucket/my-web-server-ami/  
image.manifest.xml -Name my-web-server-ami
```

- APIEinzelheiten finden Sie unter [RegisterImage AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **RejectVpcPeeringConnection** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `RejectVpcPeeringConnection`.

CLI

AWS CLI

Um eine VPC Peering-Verbindung abzulehnen

In diesem Beispiel wird die angegebene VPC Peering-Verbindungsanforderung zurückgewiesen.

Befehl:

```
aws ec2 reject-vpc-peering-connection --vpc-peering-connection-id pcx-1a2b3c4d
```


Ausgabe:

```
{
  "Return": true
}
```

- API-Einheiten finden Sie [RejectVpcPeeringConnection](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Das obige Beispiel lehnt die Anfrage nach der Anforderungs-ID VpcPeering pcx-01a2b3ce45fe67eb8 ab

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- API-Einheiten [RejectVpcPeeringConnection](#) finden AWS Tools for PowerShell Sie unter Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **ReleaseAddress** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ReleaseAddress`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Release an Elastic IP address. After the Elastic IP address is released,
/// it can no longer be used.
/// </summary>
/// <param name="allocationId">The allocation Id of the Elastic IP address.</
param>
/// <returns>True if successful.</returns>
public async Task<bool> ReleaseAddress(string allocationId)
{
    try
    {
        var request = new ReleaseAddressRequest { AllocationId =
allocationId };

        var response = await _amazonEC2.ReleaseAddressAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidAllocationID.NotFound")
        {
            _logger.LogError(
                $"AllocationId {allocationId} was not found.
{ec2Exception.Message}");
        }

        return false;
    }
    catch (Exception ex)
    {
        _logger.LogError(
```

```

        $"An error occurred while releasing the AllocationId
{allocationId}.: {ex.Message}");
        return false;
    }
}

```

- API-Einheiten finden Sie [ReleaseAddress](#) in der AWS SDK for .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

#####
# function ec2_release_address
#
# This function releases an Elastic IP address from an Amazon Elastic Compute
# Cloud (Amazon EC2) instance.
#
# Parameters:
#     -a allocation_id - The allocation ID of the Elastic IP address to
#     release.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#
#####
function ec2_release_address() {
    local allocation_id response

    # Function to display usage information
    function usage() {
        echo "function ec2_release_address"
        echo "Releases an Elastic IP address from an Amazon Elastic Compute Cloud
(Amazon EC2) instance."
    }
}

```

```

    echo " -a allocation_id - The allocation ID of the Elastic IP address to
release."
    echo ""
}

# Parse the command-line arguments
while getopts "a:h" option; do
    case "${option}" in
        a) allocation_id="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

# Validate the input parameters
if [[ -z "$allocation_id" ]]; then
    errecho "ERROR: You must provide an allocation ID with the -a parameter."
    return 1
fi

response=$(aws ec2 release-address \
--allocation-id "$allocation_id") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports release-address operation failed."
    errecho "$response"
    return 1
}

return 0
}

```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```
#####
```

```

# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then
        errecho " The service returned an error."
    elif [ "$err_code" == 255 ]; then
        errecho " 255 is a catch-all error."
    fi

    return 0
}

```

- API-Einheiten finden Sie [ReleaseAddress](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#!/ Release an Elastic IP address.
/*!
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::EC2::releaseAddress(const Aws::String &allocationID,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::EC2::EC2Client ec2(clientConfiguration);

    Aws::EC2::Model::ReleaseAddressRequest request;
    request.SetAllocationId(allocationID);

    Aws::EC2::Model::ReleaseAddressOutcome outcome = ec2.ReleaseAddress(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to release Elastic IP address " <<
                    allocationID << ":" << outcome.GetError().GetMessage() <<
                    std::endl;
    } else {
        std::cout << "Successfully released Elastic IP address " <<
                    allocationID << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API-Einheiten finden Sie [ReleaseAddress](#) in der AWS SDK for C++ API-Referenz.

CLI

AWS CLI

Um Elastic IP-Adressen für EC2 -Classic zu veröffentlichen

In diesem Beispiel wird eine Elastic IP-Adresse für die Verwendung mit Instances in EC2 -Classic veröffentlicht. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 release-address --public-ip 198.51.100.0
```

Um eine Elastic IP-Adresse für EC2 - freizugeben VPC

In diesem Beispiel wird eine Elastic IP-Adresse zur Verwendung mit Instances in a veröffentlichtVPC. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 release-address --allocation-id eipalloc-64d5890a
```

- APIEinzelheiten finden Sie [ReleaseAddress](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**  
 * Releases an Elastic IP address asynchronously.  
 *  
 * @param allocId the allocation ID of the Elastic IP address to be released
```

```

    * @return a {@link CompletableFuture} representing the asynchronous
    operation of releasing the Elastic IP address
    */
    public CompletableFuture<ReleaseAddressResponse>
    releaseEC2AddressAsync(String allocId) {
        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

        CompletableFuture<ReleaseAddressResponse> response =
        getAsyncClient().releaseAddress(request);
        response.whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to release Elastic IP
address", ex);
            }
        });

        return response;
    }

```

- API-Einheiten finden Sie [ReleaseAddress](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import { ReleaseAddressCommand, EC2Client } from "@aws-sdk/client-ec2";

/**
 * Release an Elastic IP address.
 * @param {{ allocationId: string }} options
 */
export const main = async ({ allocationId }) => {
    const client = new EC2Client({});

```



```
const command = new ReleaseAddressCommand({
  // You can also use PublicIp, but that is for EC2 classic which is being
  retired.
  AllocationId: allocationId,
});

try {
  await client.send(command);
  console.log("Successfully released address.");
} catch (caught) {
  if (
    caught instanceof Error &&
    caught.name === "InvalidAllocationID.NotFound"
  ) {
    console.warn(`${caught.message}. Please provide a valid AllocationID.`);
  } else {
    throw caught;
  }
}
};
```

- API-Einheiten finden Sie [ReleaseAddress](#) in der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
suspend fun releaseEC2AddressSc(allocId: String?) {
  val request =
    ReleaseAddressRequest {
      allocationId = allocId
    }

  Ec2Client { region = "us-west-2" }.use { ec2 ->
    ec2.releaseAddress(request)
  }
}
```

```
        println("Successfully released Elastic IP address $allocId")
    }
}
```

- API-Einheiten finden Sie [ReleaseAddress](#) in der AWS SDK-API-Kotlin-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Elastic IP-Adresse für Instances in einer veröffentlichten VPC.

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

Beispiel 2: In diesem Beispiel wird die angegebene Elastic IP-Adresse für Instances in EC2 - Classic veröffentlicht.

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- API-Einheiten finden Sie unter [ReleaseAddress AWS Tools for PowerShell Cmdlet](#)-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
    actions using the client interface."""

    class ElasticIp:
```

```
        """Represents an Elastic IP and its associated instance."""

    def __init__(
        self, allocation_id: str, public_ip: str, instance_id: Optional[str]
= None
    ) -> None:
        """
        Initializes the ElasticIp object.

        :param allocation_id: The allocation ID of the Elastic IP.
        :param public_ip: The public IP address of the Elastic IP.
        :param instance_id: The ID of the associated EC2 instance, if any.
        """
        self.allocation_id = allocation_id
        self.public_ip = public_ip
        self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def release(self, allocation_id: str) -> None:
        """
        Releases an Elastic IP address. After the Elastic IP address is released,
it can no longer be used.
        """

```

```

        :param allocation_id: The allocation ID of the Elastic IP to release.
        :raises ClientError: If the release fails, such as when the Elastic IP
address is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None:
            logger.info(f"No Elastic IP found with allocation ID
{allocation_id}.")
            return

        try:
            self.ec2_client.release_address(AllocationId=allocation_id)
            self.elastic_ips.remove(elastic_ip) # Remove the Elastic IP from the
list
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidAddress.NotFound":
                logger.error(
                    f"Failed to release Elastic IP address {allocation_id} "
                    "because it could not be found. Verify the Elastic IP address
"
                    "and ensure it is allocated to your account in the correct
region "
                    "before attempting to release it."
                )
            raise

```

- API-Einheiten finden Sie unter [ReleaseAddress](#) Python (Boto3) API -Referenz. AWS SDK

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

# Releases an Elastic IP address from an
# Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - An Amazon EC2 instance with an associated Elastic IP address.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
#   the Elastic IP address.
# @return [Boolean] true if the Elastic IP address was released;
#   otherwise, false.
# @example
#   exit 1 unless elastic_ip_address_released?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX'
#   )
def elastic_ip_address_released?(ec2_client, allocation_id)
  ec2_client.release_address(allocation_id: allocation_id)
  true
rescue StandardError => e
  puts("Error releasing Elastic IP address: #{e.message}")
  false
end

```

- API-Einheiten finden Sie [ReleaseAddress](#) in der AWS SDK for Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

pub async fn deallocate_ip_address(&self, allocation_id: &str) -> Result<(),
EC2Error> {
    self.client

```

```

        .release_address()
        .allocation_id(allocation_id)
        .send()
        .await?;
    Ok(())
}

```

- API-Einheiten finden Sie [ReleaseAddress](#) in der AWS SDK API Rust-Referenz.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

TRY.
    lo_ec2->releaseaddress( iv_allocationid = iv_allocation_id ).
    MESSAGE 'Elastic IP address released.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- API-Einheiten finden Sie [ReleaseAddress](#) unter AWS SDK SAP ABAP API als Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **ReleaseHosts** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ReleaseHosts`.

CLI

AWS CLI

Um einen Dedicated Host von deinem Konto freizugeben

Um einen Dedicated Host von deinem Konto freizugeben. Instanzen, die sich auf dem Host befinden, müssen gestoppt oder beendet werden, bevor der Host freigegeben werden kann.

Befehl:

```
aws ec2 release-hosts --host-id=h-0029d6e3cacf1b3da
```

Ausgabe:

```
{
  "Successful": [
    "h-0029d6e3cacf1b3da"
  ],
  "Unsuccessful": []
}
```

- API-Einheiten finden Sie [ReleaseHosts](#) unter AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Host-ID h-0badafd1dcb2f3456 veröffentlicht

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

Ausgabe:

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-EC2Host (ReleaseHosts)" on target
"h-0badafd1dcb2f3456".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is
"Y"): Y
```

```

Successful                Unsuccessful
-----
{h-0badafd1dcb2f3456} {}

```

- API-Einheiten [ReleaseHosts](#) finden AWS Tools for PowerShell Sie unter Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **ReplaceIamInstanceProfileAssociation** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ReplaceIamInstanceProfileAssociation`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erstellen und Verwalten eines ausfallsicheren Services](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

/// <summary>
/// Replace the profile associated with a running instance. After the profile
is replaced, the instance
/// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
/// used to restart the Python web server.
/// </summary>
/// <param name="instanceId">The Id of the instance to update.</param>

```



```
    /// <param name="credsProfileName">The name of the new profile to associate
    with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
    for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
    credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
                    instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
        }
    }
}
```

```

        await WaitForInstanceState(instanceId, InstanceStateName.Running);
        Console.WriteLine("Instance ready.");
        Console.WriteLine($"Sending restart command to instance
{instanceId}");
        await _amazonSsm.SendCommandAsync(
            new SendCommandRequest()
            {
                InstanceIds = new List<string>() { instanceId },
                DocumentName = "AWS-RunShellScript",
                Parameters = new Dictionary<string, List<string>>()
                {
                    {
                        "commands",
                        new List<string>() { "cd / && sudo python3 server.py
80" }
                    }
                }
            });
        Console.WriteLine($"Restarted the web server on instance
{instanceId}");
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
        {
            _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
        throw;
    }
}

```

- API-Einheiten finden Sie [ReplacelamInstanceProfileAssociation](#) in der AWS SDK for .NET API-Referenz.

CLI

AWS CLI

Um ein IAM Instanzprofil für eine Instanz zu ersetzen

In diesem Beispiel wird das durch die Zuordnung dargestellte IAM Instanzprofil `iip-
assoc-060bae234aac2e7fa` mit dem genannten IAM Instanzprofil `AdminRole` ersetzt.

```
aws ec2 replace-iam-instance-profile-association \  
  --iam-instance-profile Name=AdminRole \  
  --association-id iip-assoc-060bae234aac2e7fa
```


Ausgabe:

```
{  
  "IamInstanceProfileAssociation": {  
    "InstanceId": "i-087711ddaf98f9489",  
    "State": "associating",  
    "AssociationId": "iip-assoc-0b215292fab192820",  
    "IamInstanceProfile": {  
      "Id": "AIPAJLNLDX3AMYZNNWYYAY",  
      "Arn": "arn:aws:iam::123456789012:instance-profile/AdminRole"  
    }  
  }  
}
```

- API-Einheiten finden Sie [ReplacelamInstanceProfileAssociation](#) unter AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

 Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
```

```
ec2Client.send(  
    new ReplaceIamInstanceProfileAssociationCommand({  
        AssociationId: state.instanceProfileAssociationId,  
        IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },  
    })),  
),  
);
```

- API-Einheiten finden Sie [ReplaceIamInstanceProfileAssociation](#) in der AWS SDK for JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Dieses Beispiel ersetzt das Instance-Profil einer laufenden Instance, startet die Instance neu und sendet nach dem Start einen Befehl an die Instance.

```
class AutoScalingWrapper:  
    """  
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.  
    """  
  
    def __init__(  
        self,  
        resource_prefix: str,  
        inst_type: str,  
        ami_param: str,  
        autoscaling_client: boto3.client,  
        ec2_client: boto3.client,  
        ssm_client: boto3.client,  
        iam_client: boto3.client,  
    ):  
        """
```

```
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

    def replace_instance_profile(
        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
```

Replaces the profile associated with a running instance. After the profile is replaced, the instance is rebooted to ensure that it uses the new profile. When the instance is ready, Systems Manager is used to restart the Python web server.

:param instance_id: The ID of the instance to restart.

:param new_instance_profile_name: The name of the new profile to associate with

the specified instance.

:param profile_association_id: The ID of the existing profile association for the

instance.

"""

try:

```
self.ec2_client.replace_iam_instance_profile_association(
    IamInstanceProfile={"Name": new_instance_profile_name},
    AssociationId=profile_association_id,
)
```

```
log.info(
    "Replaced instance profile for association %s with profile %s.",
    profile_association_id,
    new_instance_profile_name,
)
```

```
time.sleep(5)
```

```
self.ec2_client.reboot_instances(InstanceIds=[instance_id])
log.info("Rebooting instance %s.", instance_id)
waiter = self.ec2_client.get_waiter("instance_running")
log.info("Waiting for instance %s to be running.", instance_id)
waiter.wait(InstanceIds=[instance_id])
log.info("Instance %s is now running.", instance_id)
```

```
self.ssm_client.send_command(
    InstanceIds=[instance_id],
    DocumentName="AWS-RunShellScript",
    Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
)
```

```
log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
```

```
except ClientError as err:
```

```
log.error("Failed to replace instance profile.")
error_code = err.response["Error"]["Code"]
```

```
if error_code == "InvalidAssociationID.NotFound":
    log.error(
        f"Association ID '{profile_association_id}' does not exist."
        "Please check the association ID and try again."
    )
if error_code == "InvalidInstanceId":
    log.error(
        f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
        f"Please verify the instance ID and try again."
    )
log.error(f"Full error:\n\t{err}")
```

- API-Einheiten finden Sie unter [ReplaceInstanceProfileAssociation](#) Python (Boto3) API - Referenz. AWS SDK

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **ReplaceNetworkAclAssociation** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ReplaceNetworkAclAssociation`.

CLI

AWS CLI

Um das einem Subnetz ACL zugeordnete Netzwerk zu ersetzen

In diesem Beispiel wird das angegebene Netzwerk ACL dem Subnetz für die angegebene ACL-Netzwerkzuordnung zugeordnet.

Befehl:

```
aws ec2 replace-network-acl-association --association-id aclassoc-e5b95c8c --
network-acl-id acl-5fb85d36
```

Ausgabe:

```
{
  "NewAssociationId": "aclassoc-3999875b"
}
```

- API-Einheiten finden Sie unter [ReplaceNetworkAclAssociation AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Netzwerk ACL dem Subnetz für die angegebene ACL Netzwerkzuordnung zugeordnet.

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId
aclassoc-1a2b3c4d
```

Ausgabe:

```
aclassoc-87654321
```

- API-Einheiten finden Sie unter [ReplaceNetworkAclAssociation AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **ReplaceNetworkAclEntry** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ReplaceNetworkAclEntry`.

CLI

AWS CLI

Um einen ACL Netzwerkeintrag zu ersetzen

Dieses Beispiel ersetzt einen Eintrag für das angegebene NetzwerkACL. Die neue Regel 100 erlaubt eingehenden Verkehr von 203.0.113.12/24 auf UDP Port 53 () in jedes zugehörige Subnetz. DNS

Befehl:

```
aws ec2 replace-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-number 100 --protocol udp --port-range From=53,To=53 --cidr-block 203.0.113.12/24 --rule-action allow
```

- Einzelheiten finden Sie in der Befehlsreferenz. API [ReplaceNetworkAclEntry](#) AWS CLI

PowerShell

Tools für PowerShell

Beispiel 1: Dieses Beispiel ersetzt den angegebenen Eintrag für das angegebene NetzwerkACL. Die neue Regel erlaubt eingehenden Verkehr von der angegebenen Adresse zu jedem zugehörigen Subnetz.

```
Set-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber 100 -Protocol 17 -PortRange_From 53 -PortRange_To 53 -CidrBlock 203.0.113.12/24 -RuleAction allow
```

- API Einzelheiten finden Sie unter [ReplaceNetworkAclEntry AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter. [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **ReplaceRoute** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ReplaceRoute`.

CLI

AWS CLI

Um eine Route zu ersetzen

Dieses Beispiel ersetzt die angegebene Route in der angegebenen Routentabelle. Die neue Route entspricht der angegebenen CIDR und sendet den Verkehr an das angegebene Virtual Private Gateway. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 replace-route --route-table-id rtb-22574640 --destination-cidr-block 10.0.0.0/16 --gateway-id vgw-9a4cacf3
```

- API-Einheiten finden Sie [ReplaceRoute](#) unter AWS CLI Befehlsreferenz.

PowerShell**Tools für PowerShell**

Beispiel 1: Dieses Beispiel ersetzt die angegebene Route für die angegebene Routentabelle. Die neue Route sendet den angegebenen Verkehr an das angegebene Virtual Private Gateway.

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 - GatewayId vgw-1a2b3c4d
```

- API-Einheiten finden Sie unter [ReplaceRoute AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie `ReplaceRouteTableAssociation` mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ReplaceRouteTableAssociation`.

CLI**AWS CLI**

Um die einem Subnetz zugeordnete Routentabelle zu ersetzen

In diesem Beispiel wird die angegebene Routing-Tabelle dem Subnetz für die angegebene Routentabellenzuordnung zugeordnet.

Befehl:

```
aws ec2 replace-route-table-association --association-id rtbassoc-781d0d1a --  
route-table-id rtb-22574640
```

Ausgabe:

```
{  
  "NewAssociationId": "rtbassoc-3a1f0f58"  
}
```

- API-Einheiten finden Sie unter [ReplaceRouteTableAssociation AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Routing-Tabelle dem Subnetz für die angegebene Routentabellenzuordnung zugeordnet.

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId  
rtbassoc-12345678
```

Ausgabe:

```
rtbassoc-87654321
```

- API-Einheiten finden Sie unter [ReplaceRouteTableAssociation AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **ReportInstanceState** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ReportInstanceState`.

CLI

AWS CLI

Um Statusfeedback für eine Instanz zu melden

Dieser Beispielbefehl meldet Statusfeedback für die angegebene Instanz.

Befehl:

```
aws ec2 report-instance-status --instances i-1234567890abcdef0 --status impaired
--reason-codes unresponsive
```

- API-Einheiten finden Sie [ReportInstanceStatus](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird Statusfeedback für die angegebene Instanz gemeldet.

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode
unresponsive
```

- API-Einheiten finden Sie unter [ReportInstanceStatus AWS Tools for PowerShell Cmdlet](#)-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **RequestSpotFleet** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `RequestSpotFleet`.

CLI

AWS CLI

Um eine Spot-Flotte im Subnetz mit dem niedrigsten Preis anzufordern

Dieser Beispielbefehl erstellt eine Spot-Flottenanforderung mit zwei Startspezifikationen, die sich nur je nach Subnetz unterscheiden. Die Spot-Flotte startet die Instances im angegebenen Subnetz mit dem niedrigsten Preis. Wenn die Instances standardmäßig gestartet werdenVPC, erhalten sie standardmäßig eine öffentliche IP-Adresse. Wenn die Instances nicht standardmäßig gestartet werdenVPC, erhalten sie standardmäßig keine öffentliche IP-Adresse.

Beachten Sie, dass Sie in einer Spot-Flottenanfrage nicht verschiedene Subnetze aus derselben Availability Zone angeben können.

Befehl:

```
aws ec2 request-spot-fleet --spot-fleet-request-config file://config.json
```

config.json:

```
{
  "SpotPrice": "0.04",
  "TargetCapacity": 2,
  "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role",
  "LaunchSpecifications": [
    {
      "ImageId": "ami-1a2b3c4d",
      "KeyName": "my-key-pair",
      "SecurityGroups": [
        {
          "GroupId": "sg-1a2b3c4d"
        }
      ],
      "InstanceType": "m3.medium",
      "SubnetId": "subnet-1a2b3c4d, subnet-3c4d5e6f",
      "IamInstanceProfile": {
        "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
      }
    }
  ]
}
```

Ausgabe:

```
{
```

```
"SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE"  
}
```

Um eine Spot-Flotte in der Availability Zone mit dem niedrigsten Preis anzufordern

Mit diesem Beispielbefehl wird eine Spot-Flottenanforderung mit zwei Startspezifikationen erstellt, die sich nur je nach Availability Zone unterscheiden. Die Spot-Flotte startet die Instances in der angegebenen Availability Zone mit dem niedrigsten Preis. Wenn Ihr Konto VPC nur EC2 — unterstützt, EC2 startet Amazon die Spot-Instances im Standardsubnetz der Availability Zone. Wenn Ihr Konto EC2 -Classic unterstützt, EC2 startet Amazon die Instances in EC2 -Classic in der Availability Zone.

Befehl:

```
aws ec2 request-spot-fleet --spot-fleet-request-config file://config.json
```

config.json:

```
{  
  "SpotPrice": "0.04",  
  "TargetCapacity": 2,  
  "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role",  
  "LaunchSpecifications": [  
    {  
      "ImageId": "ami-1a2b3c4d",  
      "KeyName": "my-key-pair",  
      "SecurityGroups": [  
        {  
          "GroupId": "sg-1a2b3c4d"  
        }  
      ],  
      "InstanceType": "m3.medium",  
      "Placement": {  
        "AvailabilityZone": "us-west-2a, us-west-2b"  
      },  
      "IamInstanceProfile": {  
        "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"  
      }  
    }  
  ]  
}
```

Um Spot-Instances in einem Subnetz zu starten und ihnen öffentliche IP-Adressen zuzuweisen

Dieser Beispielbefehl weist Instances, die nicht standardmäßig gestartet wurden, öffentliche Adressen zu. VPC Beachten Sie, dass Sie bei der Angabe einer Netzwerkschnittstelle die Subnetz-ID und die Sicherheitsgruppen-ID über die Netzwerkschnittstelle angeben müssen.

Befehl:

```
aws ec2 request-spot-fleet --spot-fleet-request-config file://config.json
```

config.json:

```
{
  "SpotPrice": "0.04",
  "TargetCapacity": 2,
  "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role",
  "LaunchSpecifications": [
    {
      "ImageId": "ami-1a2b3c4d",
      "KeyName": "my-key-pair",
      "InstanceType": "m3.medium",
      "NetworkInterfaces": [
        {
          "DeviceIndex": 0,
          "SubnetId": "subnet-1a2b3c4d",
          "Groups": [ "sg-1a2b3c4d" ],
          "AssociatePublicIpAddress": true
        }
      ],
      "IamInstanceProfile": {
        "Arn": "arn:aws:iam::880185128111:instance-profile/my-iam-role"
      }
    }
  ]
}
```

Um eine Spot-Flotte unter Verwendung der diversifizierten Zuweisungsstrategie anzufordern

Dieser Beispielbefehl erstellt eine Spot-Flottenanforderung, die 30 Instances unter Verwendung der diversifizierten Zuweisungsstrategie startet. Die Startspezifikationen unterscheiden sich je nach Instance-Typ. Die Spot-Flotte verteilt die Instances auf die Startspezifikationen, sodass es von jedem Typ 10 Instances gibt.

Befehl:

```
aws ec2 request-spot-fleet --spot-fleet-request-config file://config.json
```

config.json:

```
{
  "SpotPrice": "0.70",
  "TargetCapacity": 30,
  "AllocationStrategy": "diversified",
  "IamFleetRole": "arn:aws:iam::123456789012:role/my-spot-fleet-role",
  "LaunchSpecifications": [
    {
      "ImageId": "ami-1a2b3c4d",
      "InstanceType": "c4.2xlarge",
      "SubnetId": "subnet-1a2b3c4d"
    },
    {
      "ImageId": "ami-1a2b3c4d",
      "InstanceType": "m3.2xlarge",
      "SubnetId": "subnet-1a2b3c4d"
    },
    {
      "ImageId": "ami-1a2b3c4d",
      "InstanceType": "r3.2xlarge",
      "SubnetId": "subnet-1a2b3c4d"
    }
  ]
}
```

Weitere Informationen finden Sie unter Spot Fleet Requests im Amazon Elastic Compute Cloud-Benutzerhandbuch.

- API-Einheiten finden Sie [RequestSpotFleet](#) in der AWS CLI Befehlsreferenz.

PowerShell**Tools für PowerShell**

Beispiel 1: In diesem Beispiel wird eine Spot-Flottenanfrage in der Availability Zone mit dem niedrigsten Preis für den angegebenen Instance-Typ erstellt. Wenn Ihr Konto VPC nur EC2 — unterstützt, startet die Spot-Flotte die Instances in der Availability Zone mit dem niedrigsten

Preis, die über ein Standardsubnetz verfügt. Wenn Ihr Konto EC2 -Classic unterstützt, startet die Spot-Flotte die Instances in EC2 -Classic in der Availability Zone mit dem niedrigsten Preis. Beachten Sie, dass der Preis, den Sie zahlen, den angegebenen Spot-Preis für die Anfrage nicht überschreiten wird.

```
$sg = New-Object Amazon.EC2.Model.GroupIdentifier
$sg.GroupId = "sg-12345678"
$lc = New-Object Amazon.EC2.Model.SpotFleetLaunchSpecification
$lc.ImageId = "ami-12345678"
$lc.InstanceType = "m3.medium"
$lc.SecurityGroups.Add($sg)
Request-EC2SpotFleet -SpotFleetRequestConfig_SpotPrice 0.04 `
-SpotFleetRequestConfig_TargetCapacity 2 `
-SpotFleetRequestConfig_IamFleetRole arn:aws:iam::123456789012:role/my-spot-
fleet-role `
-SpotFleetRequestConfig_LaunchSpecification $lc
```

- API Einzelheiten finden Sie unter [RequestSpotFleet AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **RequestSpotInstances** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `RequestSpotInstances`.

CLI

AWS CLI

Um Spot-Instances anzufordern

Dieser Beispielbefehl erstellt eine einmalige Spot-Instance-Anfrage für fünf Instances in der angegebenen Availability Zone. Wenn Ihr Konto VPC nur EC2 — unterstützt, EC2 startet Amazon die Instances im Standardsubnetz der angegebenen Availability Zone. Wenn Ihr Konto EC2 -Classic unterstützt, EC2 startet Amazon die Instances in EC2 -Classic in der angegebenen Availability Zone.

Befehl:

```
aws ec2 request-spot-instances --spot-price "0.03" --instance-count 5 --  
type "one-time" --launch-specification file://specification.json
```

Specification.json:

```
{  
  "ImageId": "ami-1a2b3c4d",  
  "KeyName": "my-key-pair",  
  "SecurityGroupIds": [ "sg-1a2b3c4d" ],  
  "InstanceType": "m3.medium",  
  "Placement": {  
    "AvailabilityZone": "us-west-2a"  
  },  
  "IamInstanceProfile": {  
    "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"  
  }  
}
```

Ausgabe:

```
{  
  "SpotInstanceRequests": [  
    {  
      "Status": {  
        "UpdateTime": "2014-03-25T20:54:21.000Z",  
        "Code": "pending-evaluation",  
        "Message": "Your Spot request has been submitted for review, and is  
pending evaluation."  
      },  
      "ProductDescription": "Linux/UNIX",  
      "SpotInstanceRequestId": "sir-df6f405d",  
      "State": "open",  
      "LaunchSpecification": {  
        "Placement": {  
          "AvailabilityZone": "us-west-2a"  
        },  
        "ImageId": "ami-1a2b3c4d",  
        "KeyName": "my-key-pair",  
        "SecurityGroups": [  
          {  
            "GroupName": "my-security-group",  
            "GroupId": "sg-1a2b3c4d"  
          }  
        ]  
      }  
    }  
  ]  
}
```

```

        }
      ],
      "Monitoring": {
        "Enabled": false
      },
      "IamInstanceProfile": {
        "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
      },
      "InstanceType": "m3.medium"
    },
    "Type": "one-time",
    "CreateTime": "2014-03-25T20:54:20.000Z",
    "SpotPrice": "0.050000"
  },
  ...
]
}

```

Dieser Beispielbefehl erstellt eine einmalige Spot-Instance-Anfrage für fünf Instances im angegebenen Subnetz. Amazon EC2 startet die Instances im angegebenen Subnetz. Wenn VPC es sich nicht um eine Standardadresse handeltVPC, erhalten die Instances standardmäßig keine öffentliche IP-Adresse.

Befehl:

```
aws ec2 request-spot-instances --spot-price "0.050" --instance-count 5 --
type "one-time" --launch-specification file://specification.json
```

Specification.json:

```

{
  "ImageId": "ami-1a2b3c4d",
  "SecurityGroupIds": [ "sg-1a2b3c4d" ],
  "InstanceType": "m3.medium",
  "SubnetId": "subnet-1a2b3c4d",
  "IamInstanceProfile": {
    "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
  }
}

```

Ausgabe:

```
{
  "SpotInstanceRequests": [
    {
      "Status": {
        "UpdateTime": "2014-03-25T22:21:58.000Z",
        "Code": "pending-evaluation",
        "Message": "Your Spot request has been submitted for review, and is
pending evaluation."
      },
      "ProductDescription": "Linux/UNIX",
      "SpotInstanceRequestId": "sir-df6f405d",
      "State": "open",
      "LaunchSpecification": {
        "Placement": {
          "AvailabilityZone": "us-west-2a"
        }
        "ImageId": "ami-1a2b3c4d"
        "SecurityGroups": [
          {
            "GroupName": "my-security-group",
            "GroupID": "sg-1a2b3c4d"
          }
        ]
        "SubnetId": "subnet-1a2b3c4d",
        "Monitoring": {
          "Enabled": false
        },
        "IamInstanceProfile": {
          "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
        },
        "InstanceType": "m3.medium",
      },
      "Type": "one-time",
      "CreateTime": "2014-03-25T22:21:58.000Z",
      "SpotPrice": "0.050000"
    },
    ...
  ]
}
```

In diesem Beispiel wird den Spot-Instances, die Sie nicht standardmäßig starten, eine öffentliche IP-Adresse zugewiesen. VPC Beachten Sie, dass Sie bei der Angabe

einer Netzwerkschnittstelle die Subnetz-ID und die Sicherheitsgruppen-ID über die Netzwerkschnittstelle angeben müssen.

Befehl:

```
aws ec2 request-spot-instances --spot-price "0.050" --instance-count 1 --  
type "one-time" --launch-specification file://specification.json
```

Specification.json:

```
{  
  "ImageId": "ami-1a2b3c4d",  
  "KeyName": "my-key-pair",  
  "InstanceType": "m3.medium",  
  "NetworkInterfaces": [  
    {  
      "DeviceIndex": 0,  
      "SubnetId": "subnet-1a2b3c4d",  
      "Groups": [ "sg-1a2b3c4d" ],  
      "AssociatePublicIpAddress": true  
    }  
  ],  
  "IamInstanceProfile": {  
    "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"  
  }  
}
```

- API-Einheiten finden Sie [RequestSpotInstances](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine einmalige Spot-Instance im angegebenen Subnetz angefordert. Beachten Sie, dass die Sicherheitsgruppe für das erstellt werden muss VPC, das das angegebene Subnetz enthält, und dass sie über die Netzwerkschnittstelle anhand der ID angegeben werden muss. Wenn Sie eine Netzwerkschnittstelle angeben, müssen Sie die Subnetz-ID mithilfe der Netzwerkschnittstelle angeben.

```
$n = New-Object Amazon.EC2.Model.InstanceNetworkInterfaceSpecification
```

```

$n.DeviceIndex = 0
$n.SubnetId = "subnet-12345678"
$n.Groups.Add("sg-12345678")
Request-EC2SpotInstance -InstanceCount 1 -SpotPrice 0.050 -Type one-time `
-IamInstanceProfile_Arn arn:aws:iam::123456789012:instance-profile/my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType m3.medium `
-LaunchSpecification_NetworkInterface $n

```

Ausgabe:

```

ActualBlockHourlyPrice      :
AvailabilityZoneGroup       :
BlockDurationMinutes        : 0
CreateTime                  : 12/26/2015 7:44:10 AM
Fault                       :
InstanceId                  :
LaunchedAvailabilityZone    :
LaunchGroup                 :
LaunchSpecification         : Amazon.EC2.Model.LaunchSpecification
ProductDescription          : Linux/UNIX
SpotInstanceRequestId       : sir-12345678
SpotPrice                   : 0.050000
State                       : open
Status                      : Amazon.EC2.Model.SpotInstanceStatus
Tags                       : {}
Type                       : one-time

```

- API-Einheiten finden Sie unter [RequestSpotInstances AWS Tools for PowerShell Cmdlet-Referenz](#).

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **ResetImageAttribute** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ResetImageAttribute`.

CLI

AWS CLI

Um das `launchPermission` Attribut zurückzusetzen

In diesem Beispiel wird das `launchPermission` Attribut für das angegebene Attribut AMI auf seinen Standardwert zurückgesetzt. Standardmäßig AMIs sind sie privat. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 reset-image-attribute --image-id ami-5731123e --  
attribute launchPermission
```

- API-Einheiten finden Sie [ResetImageAttribute](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Attribut 'launchPermission' auf seinen Standardwert zurückgesetzt. Standardmäßig AMIs sind sie privat.

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- API-Einheiten finden Sie unter [ResetImageAttribute AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **ResetInstanceAttribute** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ResetInstanceAttribute`.

CLI

AWS CLI

Um das `sourceDestCheck` Attribut zurückzusetzen

In diesem Beispiel wird das `sourceDestCheck` Attribut der angegebenen Instanz zurückgesetzt. Die Instanz muss sich in einer VPC befinden. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --  
attribute sourceDestCheck
```

Um das `kernel`-Attribut zurückzusetzen

In diesem Beispiel wird das `kernel` Attribut der angegebenen Instanz zurückgesetzt. Die Instance muss sich im Status `stopped` befinden. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --  
attribute kernel
```

Um das `Ramdisk`-Attribut zurückzusetzen

In diesem Beispiel wird das `ramdisk` Attribut der angegebenen Instanz zurückgesetzt. Die Instance muss sich im Status `stopped` befinden. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --  
attribute ramdisk
```

- API-Einheiten finden Sie [ResetInstanceAttribute](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das Attribut 'sriovNetSupport' für die angegebene Instanz zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

Beispiel 2: In diesem Beispiel wird das Attribut 'ebsOptimized' für die angegebene Instanz zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

Beispiel 3: In diesem Beispiel wird das Attribut 'sourceDestCheck' für die angegebene Instanz zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

Beispiel 4: In diesem Beispiel wird das Attribut 'disableApiTermination' für die angegebene Instanz zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute  
disableApiTermination
```

Beispiel 5: In diesem Beispiel wird das Attribut 'instanceInitiatedShutdownBehavior' für die angegebene Instanz zurückgesetzt.

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute  
instanceInitiatedShutdownBehavior
```

- API-Einheiten finden Sie unter [ResetInstanceAttribute](#) Cmdlet-Referenz. AWS Tools for PowerShell

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **ResetNetworkInterfaceAttribute** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ResetNetworkInterfaceAttribute`.

CLI

AWS CLI

Um ein Netzwerkschnittstellenattribut zurückzusetzen

Im folgenden `reset-network-interface-attribute` Beispiel wird der Wert des Attributs für die Quell-/Zielprüfung auf zurückgesetzt. `true`

```
aws ec2 reset-network-interface-attribute \  
  --network-interface-id eni-686ea200 \  
  --source-dest-check
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- API-Einheiten finden Sie unter Befehlsreferenz [ResetNetworkInterfaceAttribute](#). AWS CLI

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Quell-/Zielüberprüfung für die angegebene Netzwerkschnittstelle zurückgesetzt.

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -  
SourceDestCheck
```

- API-Einheiten finden Sie unter [ResetNetworkInterfaceAttribute](#) Cmdlet-Referenz. AWS Tools for PowerShell

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **ResetSnapshotAttribute** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `ResetSnapshotAttribute`.

CLI

AWS CLI

Um ein Snapshot-Attribut zurückzusetzen

In diesem Beispiel werden die Berechtigungen zum Erstellen eines Volumes für den Snapshot `snap-1234567890abcdef0` zurückgesetzt. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 reset-snapshot-attribute --snapshot-id snap-1234567890abcdef0 --  
attribute createVolumePermission
```

- API-Einheiten finden Sie [ResetSnapshotAttribute](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird das angegebene Attribut des angegebenen Snapshots zurückgesetzt.

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute  
CreateVolumePermission
```

- API-Einheiten finden Sie unter [ResetSnapshotAttribute AWS Tools for PowerShell Cmdlet](#)-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **RevokeSecurityGroupEgress** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `RevokeSecurityGroupEgress`.

CLI

AWS CLI

Beispiel 1: Um die Regel zu entfernen, die ausgehenden Verkehr in einen bestimmten Adressbereich zulässt

Mit dem folgenden `revoke-security-group-egress` Beispielbefehl wird die Regel entfernt, die Zugriff auf die angegebenen Adressbereiche an TCP Port 80 gewährt.

```
aws ec2 revoke-security-group-egress \  
  --group-id sg-026c12253ce15eff7 \  
  --ip-  
permissions [[{IpProtocol=tcp,FromPort=80,ToPort=80,IpRanges=[[{CidrIp=10.0.0.0/16}]]
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Sicherheitsgruppen](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 2: Um die Regel zu entfernen, die ausgehenden Datenverkehr zu einer bestimmten Sicherheitsgruppe zulässt

Mit dem folgenden `revoke-security-group-egress` Beispielbefehl wird die Regel entfernt, die Zugriff auf die angegebene Sicherheitsgruppe an TCP Port 80 gewährt.

```
aws ec2 revoke-security-group-egress \  
  --group-id sg-026c12253ce15eff7 \  
  --ip-permissions '[[{"IpProtocol": "tcp", "FromPort": 443, "ToPort":  
443, "UserIdGroupPairs": [{"GroupId": "sg-06df23a01ff2df86d"}]]]'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Sicherheitsgruppen](#) im EC2Amazon-Benutzerhandbuch.

- API-Einheiten finden Sie [RevokeSecurityGroupEgress](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Regel für die angegebene Sicherheitsgruppe für EC2 - entfernt VPC. Dadurch wird der Zugriff auf den angegebenen IP-Adressbereich an TCP Port 80 aufgehoben. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80";  
  IpRanges="203.0.113.0/24" }  
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um das Objekt zu erstellen. IpPermission

```
$ip = New-Object Amazon.EC2.Model.IpPermission  
$ip.IpProtocol = "tcp"  
$ip.FromPort = 80  
$ip.ToPort = 80  
$ip.IpRanges.Add("203.0.113.0/24")  
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 3: In diesem Beispiel wird der Zugriff auf die angegebene Quellsicherheitsgruppe auf TCP Port 80 gesperrt.

```
$ug = New-Object Amazon.EC2.Model.UserIdGroupPair  
$ug.GroupId = "sg-1a2b3c4d"  
$ug.UserId = "123456789012"  
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission  
@( @{ IpProtocol="tcp"; FromPort="80"; ToPort="80"; UserIdGroupPairs=$ug } )
```

- API-Einheiten finden Sie unter [RevokeSecurityGroupEgress AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie `RevokeSecurityGroupIngress` mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `RevokeSecurityGroupIngress`.

CLI

AWS CLI

Beispiel 1: Um eine Regel aus einer Sicherheitsgruppe zu entfernen

Im folgenden `revoke-security-group-ingress` Beispiel wird der Zugriff auf TCP Port 22 für den `203.0.113.0/24` Adressbereich standardmäßig aus der angegebenen Sicherheitsgruppe entfernt VPC.

```
aws ec2 revoke-security-group-ingress \  
  --group-name mySecurityGroup \  
  --protocol tcp \  
  --port 22 \  
  --cidr 203.0.113.0/24
```

Dieser Befehl erzeugt keine Ausgabe, wenn er erfolgreich ist.

Weitere Informationen finden Sie unter [Sicherheitsgruppen](#) im EC2 Amazon-Benutzerhandbuch.

Beispiel 2: Um eine Regel mithilfe des IP-Berechtigungssatzes zu entfernen

Im folgenden `revoke-security-group-ingress` Beispiel wird der `ip-permissions` Parameter verwendet, um eine eingehende Regel zu entfernen, die die ICMP Nachricht zulässt `Destination Unreachable: Fragmentation Needed and Don't Fragment was Set` (Typ 3, Code 4).

```
aws ec2 revoke-security-group-ingress \  
  --group-id sg-026c12253ce15eff7 \  
  --ip-  
permissions IpProtocol=icmp,FromPort=3,ToPort=4,IpRanges=[{CidrIp=0.0.0.0/0}]
```

Dieser Befehl erzeugt keine Ausgabe, wenn er erfolgreich ist.

Weitere Informationen finden Sie unter [Sicherheitsgruppen](#) im EC2 Amazon-Benutzerhandbuch.

- API Einzelheiten finden Sie [RevokeSecurityGroupIngress](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird der Zugriff auf TCP Port 22 aus dem angegebenen Adressbereich für die angegebene Sicherheitsgruppe für EC2 - VPC gesperrt. Beachten Sie, dass Sie Sicherheitsgruppen für EC2 - anhand der Sicherheitsgruppen-ID und nicht VPC anhand des Sicherheitsgruppennamens identifizieren müssen. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";  
  IpRanges="203.0.113.0/24" }  
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 2: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um das Objekt zu erstellen. IpPermission

```
$ip = New-Object Amazon.EC2.Model.IpPermission  
$ip.IpProtocol = "tcp"  
$ip.FromPort = 22  
$ip.ToPort = 22  
$ip.IpRanges.Add("203.0.113.0/24")  
  
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

Beispiel 3: In diesem Beispiel wird der Zugriff auf TCP Port 22 aus dem angegebenen Adressbereich für die angegebene Sicherheitsgruppe für -Classic gesperrt. EC2 Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";  
  IpRanges="203.0.113.0/24" }  
  
Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

Beispiel 4: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um das Objekt zu erstellen. IpPermission

```
$ip = New-Object Amazon.EC2.Model.IpPermission  
$ip.IpProtocol = "tcp"  
$ip.FromPort = 22
```

```
$ip.ToPort = 22
$ip.IpRanges.Add("203.0.113.0/24")
```

```
Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

- API Einzelheiten finden Sie unter [RevokeSecurityGroupIngress AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter. [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **RunInstances** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `RunInstances`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Create and run an EC2 instance.
/// </summary>
/// <param name="ImageId">The image Id of the image used as a basis for the
/// EC2 instance.</param>
/// <param name="instanceType">The instance type of the EC2 instance to
create.</param>
/// <param name="keyName">The name of the key pair to associate with the
/// instance.</param>
```



```
    /// <param name="groupId">The Id of the Amazon EC2 security group that will
    be
    /// allowed to interact with the new EC2 instance.</param>
    /// <returns>The instance Id of the new EC2 instance.</returns>
    public async Task<string> RunInstances(string imageId, string instanceType,
    string keyName, string groupId)
    {
        try
        {
            var request = new RunInstancesRequest
            {
                ImageId = imageId,
                InstanceType = instanceType,
                KeyName = keyName,
                MinCount = 1,
                MaxCount = 1,
                SecurityGroupIds = new List<string> { groupId }
            };
            var response = await _amazonEC2.RunInstancesAsync(request);
            var instanceId = response.Reservation.Instances[0].InstanceId;

            Console.WriteLine("Waiting for the instance to start.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);

            return instanceId;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidGroupId.NotFound")
            {
                _logger.LogError(
                    $"GroupId {groupId} was not found. {ec2Exception.Message}");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while running the instance.: {ex.Message}");
            throw;
        }
    }
}
```

- API-Einheiten finden Sie [RunInstances](#) in der AWS SDK for .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#####
# function ec2_run_instances
#
# This function launches one or more Amazon Elastic Compute Cloud (Amazon EC2)
# instances.
#
# Parameters:
#   -i image_id - The ID of the Amazon Machine Image (AMI) to use.
#   -t instance_type - The instance type to use (e.g., t2.micro).
#   -k key_pair_name - The name of the key pair to use.
#   -s security_group_id - The ID of the security group to use.
#   -c count - The number of instances to launch (default: 1).
#   -h - Display help.
#
# Returns:
#   0 - If successful.
#   1 - If it fails.
#####
function ec2_run_instances() {
    local image_id instance_type key_pair_name security_group_id count response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_run_instances"
        echo "Launches one or more Amazon Elastic Compute Cloud (Amazon EC2)
instances."
        echo "  -i image_id - The ID of the Amazon Machine Image (AMI) to use."
    }
}
```

```
    echo " -t instance_type - The instance type to use (e.g., t2.micro)."  
    echo " -k key_pair_name - The name of the key pair to use."  
    echo " -s security_group_id - The ID of the security group to use."  
    echo " -c count - The number of instances to launch (default: 1)."  
    echo " -h - Display help."  
    echo ""  
}  
  
# Retrieve the calling parameters.  
while getopts "i:t:k:s:c:h" option; do  
    case "${option}" in  
        i) image_id="${OPTARG}" ;;  
        t) instance_type="${OPTARG}" ;;  
        k) key_pair_name="${OPTARG}" ;;  
        s) security_group_id="${OPTARG}" ;;  
        c) count="${OPTARG}" ;;  
        h)  
            usage  
            return 0  
            ;;  
        \?)  
            echo "Invalid parameter"  
            usage  
            return 1  
            ;;  
    esac  
done  
export OPTIND=1  
  
if [[ -z "$image_id" ]]; then  
    errecho "ERROR: You must provide an Amazon Machine Image (AMI) ID with the -i  
parameter."  
    usage  
    return 1  
fi  
  
if [[ -z "$instance_type" ]]; then  
    errecho "ERROR: You must provide an instance type with the -t parameter."  
    usage  
    return 1  
fi  
  
if [[ -z "$key_pair_name" ]]; then  
    errecho "ERROR: You must provide a key pair name with the -k parameter."
```

```

usage
return 1
fi

if [[ -z "$security_group_id" ]]; then
    errecho "ERROR: You must provide a security group ID with the -s parameter."
    usage
    return 1
fi

if [[ -z "$count" ]]; then
    count=1
fi

response=$(aws ec2 run-instances \
    --image-id "$image_id" \
    --instance-type "$instance_type" \
    --key-name "$key_pair_name" \
    --security-group-ids "$security_group_id" \
    --count "$count" \
    --query 'Instances[*].[InstanceId]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports run-instances operation failed.$response"
    return 1
}

echo "$response"

return 0
}

```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}


```

```
#####  
# function aws_cli_error_log()  
#  
# This function is used to log the error messages from the AWS CLI.  
#  
# The function expects the following argument:  
#     $1 - The error code returned by the AWS CLI.  
#  
# Returns:  
#     0: - Success.  
#  
#####  
function aws_cli_error_log() {  
    local err_code=$1  
    errecho "Error code : $err_code"  
    if [ "$err_code" == 1 ]; then  
        errecho " One or more S3 transfers failed."  
    elif [ "$err_code" == 2 ]; then  
        errecho " Command line failed to parse."  
    elif [ "$err_code" == 130 ]; then  
        errecho " Process received SIGINT."  
    elif [ "$err_code" == 252 ]; then  
        errecho " Command syntax invalid."  
    elif [ "$err_code" == 253 ]; then  
        errecho " The system environment or configuration was invalid."  
    elif [ "$err_code" == 254 ]; then  
        errecho " The service returned an error."  
    elif [ "$err_code" == 255 ]; then  
        errecho " 255 is a catch-all error."  
    fi  
  
    return 0  
}
```

- API-Einheiten finden Sie [RunInstances](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

//! Launch an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
  \param instanceName: A name for the EC2 instance.
  \param amiId: An Amazon Machine Image (AMI) identifier.
  \param[out] instanceID: String to return the instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::runInstance(const Aws::String &instanceName,
                              const Aws::String &amiId,
                              Aws::String &instanceID,
                              const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::RunInstancesRequest runRequest;
    runRequest.SetImageId(amiId);
    runRequest.SetInstanceType(Aws::EC2::Model::InstanceType::t1_micro);
    runRequest.SetMinCount(1);
    runRequest.SetMaxCount(1);

    Aws::EC2::Model::RunInstancesOutcome runOutcome = ec2Client.RunInstances(
        runRequest);
    if (!runOutcome.IsSuccess()) {
        std::cerr << "Failed to launch EC2 instance " << instanceName <<
            " based on ami " << amiId << ":" <<
            runOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    const Aws::Vector<Aws::EC2::Model::Instance> &instances =
runOutcome.GetResult().GetInstances();

```

```
if (instances.empty()) {
    std::cerr << "Failed to launch EC2 instance " << instanceName <<
        " based on ami " << amiId << ":" <<
        runOutcome.GetError().GetMessage() << std::endl;
    return false;
}

instanceID = instances[0].GetInstanceId();

return true;
}
```

- API-Einheiten finden Sie [RunInstances](#) in der AWS SDK for C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So starten Sie eine Instance in einem Standard-Subnetz

Im folgenden `run-instances` Beispiel wird eine einzelne Instance des Typs `t2.micro` im Standard-Subnetz für die aktuelle Region gestartet und sie dem Standard-Subnetz für die Region zugeordnet. VPC Das key pair ist optional, wenn Sie nicht vorhaben, über SSH (Linux) oder RDP (Windows) eine Verbindung zu Ihrer Instance herzustellen.

```
aws ec2 run-instances \
  --image-id ami-0abcdef1234567890 \
  --instance-type t2.micro \
  --key-name MyKeyPair
```

Ausgabe:

```
{
  "Instances": [
    {
      "AmiLaunchIndex": 0,
      "ImageId": "ami-0abcdef1234567890",
      "InstanceId": "i-1231231230abcdef0",
      "InstanceType": "t2.micro",
      "KeyName": "MyKeyPair",
```

```
"LaunchTime": "2018-05-10T08:05:20.000Z",
"Monitoring": {
  "State": "disabled"
},
"Placement": {
  "AvailabilityZone": "us-east-2a",
  "GroupName": "",
  "Tenancy": "default"
},
"PrivateDnsName": "ip-10-0-0-157.us-east-2.compute.internal",
"PrivateIpAddress": "10.0.0.157",
"ProductCodes": [],
"PublicDnsName": "",
"State": {
  "Code": 0,
  "Name": "pending"
},
"StateTransitionReason": "",
"SubnetId": "subnet-04a636d18e83cfacb",
"VpcId": "vpc-1234567890abcdef0",
"Architecture": "x86_64",
"BlockDeviceMappings": [],
"ClientToken": "",
"EbsOptimized": false,
"Hypervisor": "xen",
"NetworkInterfaces": [
  {
    "Attachment": {
      "AttachTime": "2018-05-10T08:05:20.000Z",
      "AttachmentId": "eni-attach-0e325c07e928a0405",
      "DeleteOnTermination": true,
      "DeviceIndex": 0,
      "Status": "attaching"
    },
    "Description": "",
    "Groups": [
      {
        "GroupName": "MySecurityGroup",
        "GroupId": "sg-0598c7d356eba48d7"
      }
    ],
    "Ipv6Addresses": [],
    "MacAddress": "0a:ab:58:e0:67:e2",
    "NetworkInterfaceId": "eni-0c0a29997760baee7",
```



```
    "OwnerId": "123456789012",
    "PrivateDnsName": "ip-10-0-0-157.us-east-2.compute.internal",
    "PrivateIpAddress": "10.0.0.157",
    "PrivateIpAddresses": [
      {
        "Primary": true,
        "PrivateDnsName": "ip-10-0-0-157.us-
east-2.compute.internal",
        "PrivateIpAddress": "10.0.0.157"
      }
    ],
    "SourceDestCheck": true,
    "Status": "in-use",
    "SubnetId": "subnet-04a636d18e83cfacb",
    "VpcId": "vpc-1234567890abcdef0",
    "InterfaceType": "interface"
  }
],
"RootDeviceName": "/dev/xvda",
"RootDeviceType": "ebs",
"SecurityGroups": [
  {
    "GroupName": "MySecurityGroup",
    "GroupId": "sg-0598c7d356eba48d7"
  }
],
"SourceDestCheck": true,
"StateReason": {
  "Code": "pending",
  "Message": "pending"
},
"Tags": [],
"VirtualizationType": "hvm",
"CpuOptions": {
  "CoreCount": 1,
  "ThreadsPerCore": 1
},
"CapacityReservationSpecification": {
  "CapacityReservationPreference": "open"
},
"MetadataOptions": {
  "State": "pending",
  "HttpTokens": "optional",
  "HttpPutResponseHopLimit": 1,
```

```

        "HttpEndpoint": "enabled"
      }
    }
  ],
  "OwnerId": "123456789012",
  "ReservationId": "r-02a3f596d91211712"
}

```

Beispiel 2: So starten Sie eine Instance in einem nicht standardmäßigen Subnetz und fügen eine öffentliche IP-Adresse hinzu

Im folgenden `run-instances`-Beispiel wird eine öffentliche IP-Adresse für eine Instance angefordert, die Sie in einem nicht standardmäßigen Subnetz starten. Die Instance ist mit der angegebenen Sicherheitsgruppe verbunden.

```

aws ec2 run-instances \
  --image-id ami-0abcdef1234567890 \
  --instance-type t2.micro \
  --subnet-id subnet-08fc749671b2d077c \
  --security-group-ids sg-0b0384b66d7d692f9 \
  --associate-public-ip-address \
  --key-name MyKeyPair

```

Ein Beispiel für die Ausgabe von `run-instances` finden Sie in Beispiel 1.

Beispiel 3: So starten Sie eine Instance mit zusätzlichen Volumes

Das folgende `run-instances`-Beispiel verwendet eine Blockgerät-Zuweisung, die in `mapping.json` angegeben ist, um beim Start zusätzliche Volumes anzufügen. Bei der Zuordnung von Blockgeräten können EBS Volumes, Instance-Speicher-Volumes oder sowohl EBS Volumes als auch Instance-Speicher-Volumes angegeben werden.

```

aws ec2 run-instances \
  --image-id ami-0abcdef1234567890 \
  --instance-type t2.micro \
  --subnet-id subnet-08fc749671b2d077c \
  --security-group-ids sg-0b0384b66d7d692f9 \
  --key-name MyKeyPair \
  --block-device-mappings file://mapping.json

```

Inhalt von `mapping.json`. In diesem Beispiel wird `/dev/sdh` ein leeres EBS Volume mit einer Größe von 100 GiB hinzugefügt.

```
[
  {
    "DeviceName": "/dev/sdh",
    "Ebs": {
      "VolumeSize": 100
    }
  }
]
```

Inhalt von `mapping.json`. In diesem Beispiel wird `ephemeral1` als Instance-Speicher-Volume hinzugefügt.

```
[
  {
    "DeviceName": "/dev/sdc",
    "VirtualName": "ephemeral1"
  }
]
```

Ein Beispiel für die Ausgabe von `run-instances` finden Sie in Beispiel 1.

Weitere Informationen zu Blockgerätezuordnungen finden Sie unter [Blockgerätezuordnungen](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 4: So starten Sie eine Instance und fügen bei der Erstellung Tags hinzu

Im folgenden `run-instances`-Beispiel wird ein Tag mit dem Schlüssel `webserver` und dem Wert `production` zur Instance hinzugefügt. Mit dem Befehl wird außerdem jedem erstellten Volume (in diesem Fall dem EBS Root-Volume) ein Tag mit dem Schlüssel `cost-center` und dem Wert von `cc123` zugewiesen.

```
aws ec2 run-instances \
  --image-id ami-0abcdef1234567890 \
  --instance-type t2.micro \
  --count 1 \
  --subnet-id subnet-08fc749671b2d077c \
  --key-name MyKeyPair \
  --security-group-ids sg-0b0384b66d7d692f9 \
  --tag-specifications
'ResourceType=instance,Tags=[{Key=webserver,Value=production}]'
'ResourceType=volume,Tags=[{Key=cost-center,Value=cc123}]'
```

Ein Beispiel für die Ausgabe von `run-instances` finden Sie in Beispiel 1.

Beispiel 5: So starten Sie eine Instance mit Benutzerdaten

Im folgenden `run-instances`-Beispiel werden Benutzerdaten in eine Datei mit dem Namen `my_script.txt` übergeben, die ein Konfigurationsskript für Ihre Instance enthält. Das Skript wird beim Start ausgeführt.

```
aws ec2 run-instances \  
  --image-id ami-0abcdef1234567890 \  
  --instance-type t2.micro \  
  --count 1 \  
  --subnet-id subnet-08fc749671b2d077c \  
  --key-name MyKeyPair \  
  --security-group-ids sg-0b0384b66d7d692f9 \  
  --user-data file://my_script.txt
```

Ein Beispiel für die Ausgabe von `run-instances` finden Sie in Beispiel 1.

Weitere Informationen zu Instance-Benutzerdaten finden Sie unter [Arbeiten mit Instance-Benutzerdaten](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 6: So starten Sie eine Burstable Performance Instance

Im folgenden `run-instances`-Beispiel wird eine `t2.micro`-Instance mit der `unlimited`-Kreditoption gestartet. Wenn Sie eine T2-Instance starten und keinen `--credit-specification` angeben, wird standardmäßig die Kreditoption `standard` verwendet. Wenn Sie eine T3-Instance starten, ist die Standardeinstellung die Kreditoption `unlimited`.

```
aws ec2 run-instances \  
  --image-id ami-0abcdef1234567890 \  
  --instance-type t2.micro \  
  --count 1 \  
  --subnet-id subnet-08fc749671b2d077c \  
  --key-name MyKeyPair \  
  --security-group-ids sg-0b0384b66d7d692f9 \  
  --credit-specification CpuCredits=unlimited
```

Ein Beispiel für die Ausgabe von `run-instances` finden Sie in Beispiel 1.

Weitere Informationen zu Burstable Performance Instances finden Sie unter [Burstable Performance Instances](#) im EC2Amazon-Benutzerhandbuch.

- API-Einheiten finden Sie unter [RunInstances AWS CLI-Befehlsreferenz](#).

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Runs an EC2 instance asynchronously.
 *
 * @param instanceType The instance type to use for the EC2 instance.
 * @param keyName The name of the key pair to associate with the EC2
instance.
 * @param groupName The name of the security group to associate with the EC2
instance.
 * @param amiId The ID of the Amazon Machine Image (AMI) to use for the EC2
instance.
 * @return A {@link CompletableFuture} that completes with the ID of the
started EC2 instance.
 * @throws RuntimeException If there is an error running the EC2 instance.
 */
public CompletableFuture<String> runInstanceAsync(String instanceType, String
keyName, String groupName, String amiId) {
    RunInstancesRequest runRequest = RunInstancesRequest.builder()
        .instanceType(instanceType)
        .keyName(keyName)
        .securityGroups(groupName)
        .maxCount(1)
        .minCount(1)
        .imageId(amiId)
        .build();

    CompletableFuture<RunInstancesResponse> responseFuture =
getAsyncClient().runInstances(runRequest);
    return responseFuture.thenCompose(response -> {
        String instanceIdVal = response.instances().get(0).instanceId();
    });
}
```

```

        System.out.println("Going to start an EC2 instance and use a waiter
to wait for it to be in running state");
        return getAsyncClient().waiter()
            .waitUntilInstanceExists(r -> r.instanceIds(instanceIdVal))
            .thenCompose(waitResponse -> getAsyncClient().waiter()
                .waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal))
                .thenApply(runningResponse -> instanceIdVal));
    }).exceptionally(throwable -> {
        // Handle any exceptions that occurred during the async call
        throw new RuntimeException("Failed to run EC2 instance: " +
throwable.getMessage(), throwable);
    });
}

```

- API-Einheiten finden Sie [RunInstances](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import { EC2Client, RunInstancesCommand } from "@aws-sdk/client-ec2";

/**
 * Create new EC2 instances.
 * @param {{
 *   keyName: string,
 *   securityGroupIds: string[],
 *   imageId: string,
 *   instanceType: import('@aws-sdk/client-ec2')._InstanceType,
 *   minCount?: number,
 *   maxCount?: number }} options
 */
export const main = async ({
  keyName,
  securityGroupIds,

```

```
    imageId,
    instanceType,
    minCount = "1",
    maxCount = "1",
  }) => {
    const client = new EC2Client({});
    minCount = Number.parseInt(minCount);
    maxCount = Number.parseInt(maxCount);
    const command = new RunInstancesCommand({
      // Your key pair name.
      KeyName: keyName,
      // Your security group.
      SecurityGroupIds: securityGroupIds,
      // An Amazon Machine Image (AMI). There are multiple ways to search for AMIs.
      // For more information, see:
      // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/finding-an-ami.html
      ImageId: imageId,
      // An instance type describing the resources provided to your instance. There
      // are multiple
      // ways to search for instance types. For more information see:
      // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-
      // discovery.html
      InstanceType: instanceType,
      // Availability Zones have capacity limitations that may impact your ability
      // to launch instances.
      // The `RunInstances` operation will only succeed if it can allocate at least
      // the `MinCount` of instances.
      // However, EC2 will attempt to launch up to the `MaxCount` of instances,
      // even if the full request cannot be satisfied.
      // If you need a specific number of instances, use `MinCount` and `MaxCount`
      // set to the same value.
      // If you want to launch up to a certain number of instances, use `MaxCount`
      // and let EC2 provision as many as possible.
      // If you require a minimum number of instances, but do not want to exceed a
      // maximum, use both `MinCount` and `MaxCount`.
      MinCount: minCount,
      MaxCount: maxCount,
    });

    try {
      const { Instances } = await client.send(command);
      const instanceList = Instances.map(
        (instance) => `• ${instance.InstanceId}`,
      ).join("\n");
    }
  }
}
```

```
console.log(`Launched instances:\n${instanceList}`);
} catch (caught) {
  if (caught instanceof Error && caught.name === "ResourceCountExceeded") {
    console.warn(`${caught.message}`);
  } else {
    throw caught;
  }
}
};
```

- API-Einheiten finden Sie [RunInstances](#) in der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
suspend fun createEC2Instance(
    name: String,
    amiId: String,
): String? {
    val request =
        RunInstancesRequest {
            imageId = amiId
            instanceType = InstanceType.T1Micro
            maxCount = 1
            minCount = 1
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.runInstances(request)
        val instanceId = response.instances?.get(0)?.instanceId
        val tag =
            Tag {
                key = "Name"
                value = name
            }
    }
```



```

    }

    val requestTags =
        CreateTagsRequest {
            resources = listOf(instanceId.toString())
            tags = listOf(tag)
        }
    ec2.createTags(requestTags)
    println("Successfully started EC2 Instance $instanceId based on AMI
    $amiId")
    return instanceId
}
}

```

- API-Einheiten finden Sie [RunInstances](#) in der AWS SDK-API-Kotlin-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird eine einzelne Instanz des AMI in EC2 -Classic angegebenen Objekts oder einer VPC Standardinstanz gestartet.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType
m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

Beispiel 2: In diesem Beispiel wird eine einzelne Instanz der AMI in a VPC angegebenen Instanz gestartet.

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId
subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId
sg-12345678
```

Beispiel 3: Um ein EBS Volume oder ein Instance-Speicher-Volume hinzuzufügen, definieren Sie eine Blockgeräte-Zuordnung und fügen Sie sie dem Befehl hinzu. In diesem Beispiel wird ein Instance-Speicher-Volume hinzugefügt.

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
```

```
$bdm.DeviceName = "/dev/sdf"  
  
New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

Beispiel 4: Um eines der aktuellen Windows-Betriebssysteme anzugeben AMIs, rufen Sie dessen AMI ID mithilfe von `Get-EC2ImageByName`. In diesem Beispiel wird eine Instanz aus der aktuellen Basis AMI für Windows Server 2016 gestartet.

```
$ami = Get-EC2ImageByName WINDOWS_2016_BASE  
  
New-EC2Instance -ImageId $ami.ImageId ...
```

Beispiel 5: Startet eine Instanz in der angegebenen dedizierten Host-Umgebung.

```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair  
-SecurityGroupId sg-1a2b3c4d -AvailabilityZone us-west-1a -Tenancy host -HostID  
h-1a2b3c4d5e6f1a2b3
```


Beispiel 6: Diese Anfrage startet zwei Instances und wendet ein Tag mit dem Schlüssel `Webserver` und dem Wert `production` auf die Instanzen an. Die Anfrage wendet außerdem ein Tag mit dem Schlüssel `cost-center` und dem Wert `cc123` auf die erstellten Volumes an (in diesem Fall das Root-Volume für jede Instanz).

```
$tag1 = @{ Key="webserver"; Value="production" }  
$tag2 = @{ Key="cost-center"; Value="cc123" }  
  
$tagspec1 = new-object Amazon.EC2.Model.TagSpecification  
$tagspec1.ResourceType = "instance"  
$tagspec1.Tags.Add($tag1)  
  
$tagspec2 = new-object Amazon.EC2.Model.TagSpecification  
$tagspec2.ResourceType = "volume"  
$tagspec2.Tags.Add($tag2)  
  
New-EC2Instance -ImageId "ami-1a2b3c4d" -KeyName "my-key-pair" -MaxCount 2 -  
InstanceType "t2.large" -SubnetId "subnet-1a2b3c4d" -TagSpecification $tagspec1,  
$tagspec2
```

- API-Einheiten finden Sie unter [RunInstances AWS Tools for PowerShell Cmdlet-Referenz](#).

Python

SDK für Python (Boto3)

 Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
        objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
        EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
def create(
    self,
    image_id: str,
    instance_type: str,
    key_pair_name: str,
    security_group_ids: Optional[List[str]] = None,
) -> List[Dict[str, Any]]:
    """
    Creates a new EC2 instance in the default VPC of the current account.

    The instance starts immediately after it is created.

    :param image_id: The ID of the Amazon Machine Image (AMI) to use for the
    instance.
    :param instance_type: The type of instance to create, such as 't2.micro'.
    :param key_pair_name: The name of the key pair to use for SSH access.
    :param security_group_ids: A list of security group IDs to associate with
    the instance.

                                If not specified, the default security group
    of the VPC is used.
    :return: A list of dictionaries representing Boto3 Instance objects
    representing the newly created instances.
    """
    try:
        instance_params = {
            "ImageId": image_id,
            "InstanceType": instance_type,
            "KeyName": key_pair_name,
        }
        if security_group_ids is not None:
            instance_params["SecurityGroupIds"] = security_group_ids

        response = self.ec2_client.run_instances(
            **instance_params, MinCount=1, MaxCount=1
        )
        instance = response["Instances"][0]
        self.instances.append(instance)
        waiter = self.ec2_client.get_waiter("instance_running")
        waiter.wait(InstanceIds=[instance["InstanceId"]])
    except ClientError as err:
        params_str = "\n\t".join(
            f"{key}: {value}" for key, value in instance_params.items()
        )
```

```

        logger.error(
            f"Failed to complete instance creation request.\nRequest details:
{params_str}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "InstanceLimitExceeded":
            logger.error(
                (
                    f"Insufficient capacity for instance type
'{instance_type}'. "
                    "Terminate unused instances or contact AWS Support for a
limit increase."
                )
            )
        if error_code == "InsufficientInstanceCapacity":
            logger.error(
                (
                    f"Insufficient capacity for instance type
'{instance_type}'. "
                    "Select a different instance type or launch in a
different availability zone."
                )
            )
        raise
    return self.instances

```

- API-Einheiten finden Sie unter [RunInstances](#) Python (Boto3) API -Referenz. AWS SDK

Rust

SDK für Rust

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

pub async fn create_instance<'a>(
    &self,

```

```
    image_id: &'a str,
    instance_type: InstanceType,
    key_pair: &'a KeyPairInfo,
    security_groups: Vec<&'a SecurityGroup>,
) -> Result<String, EC2Error> {
    let run_instances = self
        .client
        .run_instances()
        .image_id(image_id)
        .instance_type(instance_type)
        .key_name(
            key_pair
                .key_name()
                .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?),
        )
        .set_security_group_ids(Some(
            security_groups
                .iter()
                .filter_map(|sg| sg.group_id.clone())
                .collect(),
        ))
        .min_count(1)
        .max_count(1)
        .send()
        .await?;

    if run_instances.instances().is_empty() {
        return Err(EC2Error::new("Failed to create instance"));
    }

    let instance_id = run_instances.instances()[0].instance_id().unwrap();
    let response = self
        .client
        .create_tags()
        .resources(instance_id)
        .tags(
            Tag::builder()
                .key("Name")
                .value("From SDK Examples")
                .build(),
        )
        .send()
        .await;
```

```

match response {
  Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
  Err(err) => {
    tracing::info!("Error applying tags to {instance_id}: {err:?}");
    return Err(err.into());
  }
}

tracing::info!("Instance is created.");

Ok(instance_id.to_string())
}

```

- API-Einheiten finden Sie [RunInstances](#) in der AWS SDK API Rust-Referenz.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

" Create tags for resource created during instance launch. "
DATA lt_tagsspecifications TYPE /aws1/
cl_ec2tagsspecification=>tt_tagsspecificationlist.
DATA ls_tagsspecifications LIKE LINE OF lt_tagsspecifications.
ls_tagsspecifications = NEW /aws1/cl_ec2tagsspecification(
  iv_resourcetype = 'instance'
  it_tags = VALUE /aws1/cl_ec2tag=>tt_taglist(
    ( NEW /aws1/cl_ec2tag( iv_key = 'Name' iv_value = iv_tag_value ) )
  )
).
APPEND ls_tagsspecifications TO lt_tagsspecifications.

TRY.
  " Create/launch Amazon Elastic Compute Cloud (Amazon EC2) instance. "

```

```

        oo_result = lo_ec2->runinstances(
is returned for testing purposes. "
        iv_imageid = iv_ami_id
        iv_instancetype = 't2.micro'
        iv_maxcount = 1
        iv_mincount = 1
        it_tagspecifications = lt_tagspecifications
        iv_subnetid = iv_subnet_id
    ).
    MESSAGE 'EC2 instance created.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- API-Einheiten finden Sie [RunInstances](#) unter AWS SDKSAPABAPI als Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **RunScheduledInstances** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `RunScheduledInstances`.

CLI

AWS CLI

Um eine geplante Instanz zu starten

In diesem Beispiel wird die angegebene Scheduled Instance in einem gestartet VPC.

Befehl:

```

aws ec2 run-scheduled-instances --scheduled-instance-
id sci-1234-1234-1234-1234-123456789012 --instance-count 1 --launch-
specification file://launch-specification.json

```

launch-specification.json:


```
{
  "ImageId": "ami-12345678",
  "KeyName": "my-key-pair",
  "InstanceType": "c4.large",
  "NetworkInterfaces": [
    {
      "DeviceIndex": 0,
      "SubnetId": "subnet-12345678",
      "AssociatePublicIpAddress": true,
      "Groups": ["sg-12345678"]
    }
  ],
  "IamInstanceProfile": {
    "Name": "my-iam-role"
  }
}
```

Ausgabe:

```
{
  "InstanceIdSet": [
    "i-1234567890abcdef0"
  ]
}
```

In diesem Beispiel wird die angegebene Scheduled Instance in -Classic gestartet. EC2

Befehl:

```
aws ec2 run-scheduled-instances --scheduled-instance-
id sci-1234-1234-1234-1234-123456789012 --instance-count 1 --launch-
specification file://launch-specification.json
```

launch-specification.json:

```
{
  "ImageId": "ami-12345678",
  "KeyName": "my-key-pair",
  "SecurityGroupIds": ["sg-12345678"],
  "InstanceType": "c4.large",
  "Placement": {
    "AvailabilityZone": "us-west-2b"
  }
}
```

```
}
  "IamInstanceProfile": {
    "Name": "my-iam-role"
  }
}
```

Ausgabe:

```
{
  "InstanceIdSet": [
    "i-1234567890abcdef0"
  ]
}
```

- API-Einheiten finden Sie in der Befehlsreferenz [RunScheduledInstances](#).AWS CLI

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Scheduled Instance gestartet.

```
New-EC2ScheduledInstance -ScheduledInstanceId
sci-1234-1234-1234-1234-123456789012 -InstanceCount 1 `
-IamInstanceProfile_Name my-iam-role `
-LaunchSpecification_ImageId ami-12345678 `
-LaunchSpecification_InstanceType c4.large `
-LaunchSpecification_SubnetId subnet-12345678 `
-LaunchSpecification_SecurityGroupId sg-12345678
```

- API-Einheiten finden Sie unter [RunScheduledInstances AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **StartInstances** mit einem AWS SDK oder CLI


Die folgenden Codebeispiele zeigen, wie man es benutzt `StartInstances`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)

.NET

AWS SDK for .NET

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Start an EC2 instance.
/// </summary>
/// <param name="ec2InstanceId">The instance Id of the Amazon EC2 instance
/// to start.</param>
/// <returns>Async task.</returns>
public async Task StartInstances(string ec2InstanceId)
{
    try
    {
        var request = new StartInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId },
        };

        await _amazonEC2.StartInstancesAsync(request);

        Console.WriteLine("Waiting for instance to start. ");
        await WaitForInstanceState(ec2InstanceId, InstanceStateName.Running);
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidInstanceId")
        {
            _logger.LogError(
```

```
        $"InstanceId is invalid, unable to start.
{ec2Exception.Message}");
    }

    throw;
}
catch (Exception ex)
{
    _logger.LogError(
        $"An error occurred while starting the instance.: {ex.Message}");
    throw;
}
}

/// <summary>
/// Wait until an EC2 instance is in a specified state.
/// </summary>
/// <param name="instanceId">The instance Id.</param>
/// <param name="stateName">The state to wait for.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
{
    var request = new DescribeInstancesRequest
    {
        InstanceIds = new List<string> { instanceId }
    };

    // Wait until the instance is in the specified state.
    var hasState = false;
    do
    {
        // Wait 5 seconds.
        Thread.Sleep(5000);

        // Check for the desired state.
        var response = await _amazonEC2.DescribeInstancesAsync(request);
        var instance = response.Reservations[0].Instances[0];
        hasState = instance.State.Name == stateName;
        Console.WriteLine(". ");
    } while (!hasState);

    return hasState;
}
```

- API-Einheiten finden Sie [StartInstances](#) in der AWS SDK for .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#####
# function ec2_start_instances
#
# This function starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
# instances.
#
# Parameters:
#   -i instance_id - The ID(s) of the instance(s) to start (comma-separated).
#   -h - Display help.
#
# Returns:
#   0 - If successful.
#   1 - If it fails.
#####
function ec2_start_instances() {
    local instance_ids
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_start_instances"
        echo "Starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
instances."
        echo "  -i instance_id - The ID(s) of the instance(s) to start (comma-
separated)."
        echo "  -h - Display help."
        echo ""
    }
}
```

```

}

# Retrieve the calling parameters.
while getopts "i:h" option; do
  case "${option}" in
    i) instance_ids="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$instance_ids" ]]; then
  errecho "ERROR: You must provide one or more instance IDs with the -i
parameter."
  usage
  return 1
fi

response=$(aws ec2 start-instances \
  --instance-ids "${instance_ids}") || {
  aws_cli_error_log ${?}
  errecho "ERROR: AWS reports start-instances operation failed with $response."
  return 1
}

return 0
}

```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).

```

```
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then
        errecho " The service returned an error."
    elif [ "$err_code" == 255 ]; then
        errecho " 255 is a catch-all error."
    fi


    return 0
}

```

- API-Einheiten finden Sie [StartInstances](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#!/ Start an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
 \param instanceID: An EC2 instance ID.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::EC2::startInstance(const Aws::String &instanceId,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::StartInstancesRequest startRequest;
    startRequest.AddInstanceIds(instanceId);
    startRequest.SetDryRun(true);

    Aws::EC2::Model::StartInstancesOutcome dryRunOutcome =
    ec2Client.StartInstances(startRequest);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to start instance. A dry run should trigger an
            error."
            << std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
                Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to start instance " << instanceId << ": "
            << dryRunOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    startRequest.SetDryRun(false);
```



```
Aws::EC2::Model::StartInstancesOutcome startInstancesOutcome =
ec2Client.StartInstances(startRequest);

if (!startInstancesOutcome.IsSuccess()) {
    std::cout << "Failed to start instance " << instanceId << ": " <<
        startInstancesOutcome.GetError().GetMessage() << std::endl;
} else {
    std::cout << "Successfully started instance " << instanceId <<
        std::endl;
}

return startInstancesOutcome.IsSuccess();
}
```

- API-Einheiten finden Sie [StartInstances](#) in der AWS SDK for C++ API-Referenz.

CLI

AWS CLI

Um eine EC2 Amazon-Instanz zu starten

In diesem Beispiel wird die angegebene EBS Amazon-gestützte Instanz gestartet.

Befehl:

```
aws ec2 start-instances --instance-ids i-1234567890abcdef0
```

Ausgabe:

```
{
  "StartingInstances": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "CurrentState": {
        "Code": 0,
        "Name": "pending"
      },
      "PreviousState": {
        "Code": 80,
        "Name": "stopped"
      }
    }
  ]
}
```

```
    }  
  }  
]  
}
```

Weitere Informationen finden Sie unter Ihre Instance anhalten und starten im Benutzerhandbuch zu Amazon Elastic Compute Cloud.

- API-Einheiten finden Sie [StartInstances](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**  
 * Starts an Amazon EC2 instance asynchronously and waits until it is in the  
 "running" state.  
 *  
 * @param instanceId the ID of the instance to start  
 * @return a {@link CompletableFuture} that completes when the instance has  
 been started and is in the "running" state, or exceptionally if an error occurs  
 */  
public CompletableFuture<Void> startInstanceAsync(String instanceId) {  
    StartInstancesRequest startRequest = StartInstancesRequest.builder()  
        .instanceIds(instanceId)  
        .build();  
  
    Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()  
        .client(getAsyncClient())  
        .build();  
  
    DescribeInstancesRequest describeRequest =  
    DescribeInstancesRequest.builder()  
        .instanceIds(instanceId)  
        .build();
```

```

    logger.info("Starting instance " + instanceId + " and waiting for it to
run.");
    CompletableFuture<Void> resultFuture = new CompletableFuture<>();
    return getAsyncClient().startInstances(startRequest)
        .thenCompose(response ->
            ec2Waiter.waitForInstanceRunning(describeRequest)
        )
        .thenAccept(waiterResponse -> {
            logger.info("Successfully started instance " + instanceId);
            resultFuture.complete(null);
        })
        .exceptionally(throwable -> {
            resultFuture.completeExceptionally(new RuntimeException("Failed
to start instance: " + throwable.getMessage(), throwable));
            return null;
        });
}

```

- API-Einheiten finden Sie [StartInstances](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import { EC2Client, StartInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Starts an Amazon EBS-backed instance that you've previously stopped.
 * @param {{ instanceIds }} options
 */
export const main = async ({ instanceIds }) => {
    const client = new EC2Client({});
    const command = new StartInstancesCommand({

```

```
    InstanceIds: instanceIds,
  });

  try {
    const { StartingInstances } = await client.send(command);
    const instanceIdList = StartingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Starting instances:");
    console.log(instanceIdList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- API-Einheiten finden Sie [StartInstances](#) in der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
```

```

    ec2.startInstances(request)
    println("Waiting until instance $instanceId starts. This will take a few
minutes.")
    ec2.waitUntilInstanceRunning {
        // suspend call
        instanceIds = listOf(instanceId)
    }
    println("Successfully started instance $instanceId")
}
}

```

- API-Einheiten finden Sie [StartInstances](#) in der AWS SDK-API-Kotlin-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instanz gestartet.

```
Start-EC2Instance -InstanceId i-12345678
```

Ausgabe:

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

Beispiel 2: In diesem Beispiel werden die angegebenen Instanzen gestartet.

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

Beispiel 3: In diesem Beispiel wird die Gruppe von Instanzen gestartet, die derzeit gestoppt sind. Die von zurückgegebenen Instanzobjekte `Get-EC2Instance` werden über die Pipeline an `Start-EC2Instance` übergeben. Die in diesem Beispiel verwendete Syntax erfordert PowerShell Version 3 oder höher.

```
(Get-EC2Instance -Filter @{ Name="instance-state-name";
Values="stopped"}).Instances | Start-EC2Instance
```

Beispiel 4: Bei PowerShell Version 2 müssen Sie New-Object verwenden, um den Filter für den Filter-Parameter zu erstellen.

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "instance-state-name"
$filter.Values = "stopped"

(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

- API-Einheiten finden Sie unter [StartInstances AWS Tools for PowerShell](#) Cmdlet-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
        objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []
```

```
@classmethod
def from_client(cls) -> "EC2InstanceWrapper":
    """
    Creates an EC2InstanceWrapper instance with a default EC2 client.

    :return: An instance of EC2InstanceWrapper initialized with the default
    EC2 client.
    """
    ec2_client = boto3.client("ec2")
    return cls(ec2_client)

def start(self) -> Optional[Dict[str, Any]]:
    """
    Starts instances and waits for them to be in a running state.

    :return: The response to the start request.
    """
    if not self.instances:
        logger.info("No instances to start.")
        return None

    instance_ids = [instance["InstanceId"] for instance in self.instances]
    try:
        start_response =
self.ec2_client.start_instances(InstanceIds=instance_ids)
        waiter = self.ec2_client.get_waiter("instance_running")
        waiter.wait(InstanceIds=instance_ids)
        return start_response
    except ClientError as err:
        logger.error(
            f"Failed to start instance(s): {' '.join(map(str,
instance_ids))}")
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "IncorrectInstanceState":
            logger.error(
                "Couldn't start instance(s) because they are in an incorrect
state. "
                "Ensure the instances are in a stopped state before starting
them."
            )
        raise
```

- API-Einheiten finden Sie unter [StartInstances](#) Python (Boto3) API -Referenz. AWS SDK

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
require 'aws-sdk-ec2'

# Attempts to start an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was started; otherwise, false.
# @example
#   exit 1 unless instance_started?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_started?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
    case state
    when 'pending'
      puts 'Error starting instance: the instance is pending. Try again later.'
      return false
    when 'running'
```



```
    puts 'The instance is already running.'
    return true
  when 'terminated'
    puts 'Error starting instance: ' \
      'the instance is terminated, so you cannot start it.'
    return false
  end
end

ec2_client.start_instances(instance_ids: [instance_id])
ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
puts 'Instance started.'
true
rescue StandardError => e
  puts "Error starting instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to start instance '#{instance_id}' " \
```

```
'(this might take a few minutes)...'  
return if instance_started?(ec2_client, instance_id)  
  
puts 'Could not start instance.'  
end  
  
run_me if $PROGRAM_NAME == __FILE__
```

- API-Einheiten finden Sie [StartInstances](#) in der AWS SDK for Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Starten Sie eine EC2-Instanz anhand der Instanz-ID.

```
pub async fn start_instance(&self, instance_id: &str) -> Result<(), EC2Error>  
{  
    tracing::info!("Starting instance {instance_id}");  
  
    self.client  
        .start_instances()  
        .instance_ids(instance_id)  
        .send()  
        .await?;  
  
    tracing::info!("Started instance.");  
  
    Ok(())  
}
```

Warten Sie mithilfe der WaitersAPI, bis sich eine Instance im Status Ready und im Status OK befindet. Die Verwendung von Waiters API erfordert `use aws_sdk_ec2: :client: :Waiters` in der Rust-Datei.

```

/// Wait for an instance to be ready and status ok (default wait 60 seconds)
pub async fn wait_for_instance_ready(
    &self,
    instance_id: &str,
    duration: Option<Duration>,
) -> Result<(), EC2Error> {
    self.client
        .wait_until_instance_status_ok()
        .instance_ids(instance_id)
        .wait(duration.unwrap_or(Duration::from_secs(60)))
        .await
        .map_err(|err| match err {
            WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                "Exceeded max time ({}s) waiting for instance to start.",
                exceeded.max_wait().as_secs()
            )),
            _ => EC2Error::from(err),
        })?;
    Ok(())
}

```

- Einzelheiten [StartInstances](#) finden AWS SDK Sie in der Rust-Referenz. API API

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.

```

```

APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
lt_instance_ids.

"Perform dry run"
TRY.
    " DryRun is set to true. This checks for the required permissions to
start the instance without actually making the request. "
    lo_ec2->startinstances(
        it_instanceids = lt_instance_ids
        iv_dryrun = abap_true
    ).
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    " If the error code returned is `DryRunOperation`, then you have the
required permissions to start this instance. "
    IF lo_exception->av_err_code = 'DryRunOperation'.
        MESSAGE 'Dry run to start instance completed.' TYPE 'I'.
        " DryRun is set to false to start instance. "
        oo_result = lo_ec2->startinstances(          " oo_result is returned
for testing purposes. "
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false
        ).
        MESSAGE 'Successfully started the EC2 instance.' TYPE 'I'.
        " If the error code returned is `UnauthorizedOperation`, then you don't
have the required permissions to start this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
            MESSAGE 'Dry run to start instance failed. User does not have
permissions to start the instance.' TYPE 'E'.
        ELSE.
            DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
            MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.
ENDTRY.

```

- API Einheiten finden Sie [StartInstances](#) unter AWS SDKSAPABAPI als Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **StopInstances** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `StopInstances`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Stop an EC2 instance.
/// </summary>
/// <param name="ec2InstanceId">The instance Id of the EC2 instance to
/// stop.</param>
/// <returns>Async task.</returns>
public async Task StopInstances(string ec2InstanceId)
{
    try
    {
        var request = new StopInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId },
        };

        await _amazonEC2.StopInstancesAsync(request);
        Console.WriteLine("Waiting for the instance to stop.");
        await WaitForInstanceState(ec2InstanceId, InstanceStateName.Stopped);

        Console.WriteLine("\nThe instance has stopped.");
    }
    catch (AmazonEC2Exception ec2Exception)
    {
```

```
        if (ec2Exception.ErrorCode == "InvalidInstanceId")
        {
            _logger.LogError(
                $"InstanceId is invalid, unable to stop.
{ec2Exception.Message}");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(
            $"An error occurred while stopping the instance.: {ex.Message}");
        throw;
    }
}

/// <summary>
/// Wait until an EC2 instance is in a specified state.
/// </summary>
/// <param name="instanceId">The instance Id.</param>
/// <param name="stateName">The state to wait for.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
{
    var request = new DescribeInstancesRequest
    {
        InstanceIds = new List<string> { instanceId }
    };

    // Wait until the instance is in the specified state.
    var hasState = false;
    do
    {
        // Wait 5 seconds.
        Thread.Sleep(5000);

        // Check for the desired state.
        var response = await _amazonEC2.DescribeInstancesAsync(request);
        var instance = response.Reservations[0].Instances[0];
        hasState = instance.State.Name == stateName;
        Console.WriteLine(". ");
    } while (!hasState);
}
```

```

    return hasState;
}

```

- API-Einheiten finden Sie [StopInstances](#) in der AWS SDK for .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

#####
# function ec2_stop_instances
#
# This function stops one or more Amazon Elastic Compute Cloud (Amazon EC2)
# instances.
#
# Parameters:
#     -i instance_id - The ID(s) of the instance(s) to stop (comma-separated).
#     -h - Display help.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_stop_instances() {
    local instance_ids
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_stop_instances"
        echo "Stops one or more Amazon Elastic Compute Cloud (Amazon EC2) instances."
        echo "  -i instance_id - The ID(s) of the instance(s) to stop (comma-
separated)."
    }
}

```

```

    echo " -h - Display help."
    echo ""
}

# Retrieve the calling parameters.
while getopts "i:h" option; do
    case "${option}" in
        i) instance_ids="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

if [[ -z "$instance_ids" ]]; then
    errecho "ERROR: You must provide one or more instance IDs with the -i
parameter."
    usage
    return 1
fi

response=$(aws ec2 stop-instances \
    --instance-ids "${instance_ids}") || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports stop-instances operation failed with $response."
    return 1
}

return 0
}

```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```

#####
# function errecho

```



```

#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then
        errecho " The service returned an error."
    elif [ "$err_code" == 255 ]; then
        errecho " 255 is a catch-all error."
    fi


    return 0
}

```

- API-Einheiten finden Sie [StopInstances](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#!/ Stop an EC2 instance.
/*!
 \param instanceID: An EC2 instance ID.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::EC2::stopInstance(const Aws::String &instanceId,
                               const Aws::Client::ClientConfiguration
                               &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::StopInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::StopInstancesOutcome dryRunOutcome =
    ec2Client.StopInstances(request);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to stop instance. A dry run should trigger an
error."
            << std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
               Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to stop instance " << instanceId << ": "
            << dryRunOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::StopInstancesOutcome outcome =
    ec2Client.StopInstances(request);
```

```
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to stop instance " << instanceId << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully stopped instance " << instanceId <<
            std::endl;
    }

    return outcome.IsSuccess();
}

void PrintUsage() {
    std::cout << "Usage: run_start_stop_instance <instance_id> <start|stop>" <<
        std::endl;
}
```

- API-Einheiten finden Sie [StopInstances](#) in der AWS SDK for C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: Um eine EC2 Amazon-Instanz zu stoppen

Das folgende `stop-instances` Beispiel stoppt die angegebene EBS Amazon-gestützte Instanz.

```
aws ec2 stop-instances \
    --instance-ids i-1234567890abcdef0
```

Ausgabe:

```
{
  "StoppingInstances": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "CurrentState": {
        "Code": 64,
        "Name": "stopping"
      },
      "PreviousState": {
```

```
        "Code": 16,  
        "Name": "running"  
      }  
    ]  
  }  
}
```

Weitere Informationen finden Sie unter [Ihre Instance anhalten und starten](#) im Benutzerhandbuch zu Amazon Elastic Compute Cloud.

Beispiel 2: Um eine Amazon-Instance in den Ruhezustand zu versetzen EC2

Im folgenden `stop-instances` Beispiel wird eine EBS Amazon-gestützte Instance in den Ruhezustand versetzt, wenn die Instance für den Ruhezustand aktiviert ist und die Voraussetzungen für den Ruhezustand erfüllt. Nachdem die Instance in den Ruhezustand versetzt wurde, wird die Instance angehalten.

```
aws ec2 stop-instances \  
  --instance-ids i-1234567890abcdef0 \  
  --hibernate
```

Ausgabe:

```
{  
  "StoppingInstances": [  
    {  
      "CurrentState": {  
        "Code": 64,  
        "Name": "stopping"  
      },  
      "InstanceId": "i-1234567890abcdef0",  
      "PreviousState": {  
        "Code": 16,  
        "Name": "running"  
      }  
    }  
  ]  
}
```

Weitere Informationen finden Sie unter [Ihre On-Demand-Linux-Instance in den Ruhezustand versetzen](#) im Benutzerhandbuch zu Amazon Elastic Cloud Compute.

- Einzelheiten finden Sie in der Befehlsreferenz. API [StopInstances](#) AWS CLI

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Stops the EC2 instance with the specified ID asynchronously and waits for
 the instance to stop.
 *
 * @param instanceId the ID of the EC2 instance to stop
 * @return a {@link CompletableFuture} that completes when the instance has
 been stopped, or exceptionally if an error occurs
 */
public CompletableFuture<Void> stopInstanceAsync(String instanceId) {
    StopInstancesRequest stopRequest = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
        .client(getAsyncClient())
        .build();

    CompletableFuture<Void> resultFuture = new CompletableFuture<>();
    logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
    getAsyncClient().stopInstances(stopRequest)
        .thenCompose(response -> {
            if (response.stoppingInstances().isEmpty()) {
```

```

        return CompletableFuture.failedFuture(new
RuntimeException("No instances were stopped. Please check the instance ID: " +
instanceId));
    }
    return ec2Waiter.waitUntilInstanceStopped(describeRequest);
})
.thenAccept(waiterResponse -> {
    logger.info("Successfully stopped instance " + instanceId);
    resultFuture.complete(null);
})
.exceptionally(throwable -> {
    logger.error("Failed to stop instance " + instanceId + ": " +
throwable.getMessage(), throwable);
    resultFuture.completeExceptionally(new RuntimeException("Failed
to stop instance: " + throwable.getMessage(), throwable));
    return null;
});
}

return resultFuture;
}

```

- API-Einheiten finden Sie [StopInstances](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

import { EC2Client, StopInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Stop one or more EC2 instances.
 * @param {{ instanceIds: string[] }} options
 */

```

```
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new StopInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { StoppingInstances } = await client.send(command);
    const instanceIdList = StoppingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Stopping instances:");
    console.log(instanceIdList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- API-Einheiten finden Sie [StopInstances](#) in der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }
}
```

```

    }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
minutes.")
        ec2.waitUntilInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}

```

- API-Einheiten finden Sie [StopInstances](#) in der AWS SDK-API-Kotlin-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instanz gestoppt.

```
Stop-EC2Instance -InstanceId i-12345678
```

Ausgabe:

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

- API-Einheiten finden Sie unter [StopInstances AWS Tools for PowerShell Cmdlet](#)-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
        objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
        EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
def stop(self) -> Optional[Dict[str, Any]]:
    """
    Stops instances and waits for them to be in a stopped state.

    :return: The response to the stop request, or None if there are no
instances to stop.
    """
    if not self.instances:
        logger.info("No instances to stop.")
        return None

    instance_ids = [instance["InstanceId"] for instance in self.instances]
    try:
        # Attempt to stop the instances
        stop_response =
self.ec2_client.stop_instances(InstanceIds=instance_ids)
        waiter = self.ec2_client.get_waiter("instance_stopped")
        waiter.wait(InstanceIds=instance_ids)
    except ClientError as err:
        logger.error(
            f"Failed to stop instance(s): {','.join(map(str, instance_ids))}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "IncorrectInstanceState":
            logger.error(
                "Couldn't stop instance(s) because they are in an incorrect
state. "
                "Ensure the instances are in a running state before stopping
them."
            )
            raise
        return stop_response
```

- API Einheiten finden Sie unter [StopInstances](#) Python (Boto3) API -Referenz. AWS SDK

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
require 'aws-sdk-ec2'

# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was stopped; otherwise, false.
# @example
#   exit 1 unless instance_stopped?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_stopped?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

  if response.instance_statuses.count.positive?
    state = response.instance_statuses[0].instance_state.name
    case state
    when 'stopping'
      puts 'The instance is already stopping.'
      return true
    when 'stopped'
      puts 'The instance is already stopped.'
      return true
    when 'terminated'
      puts 'Error stopping instance: ' \
        'the instance is terminated, so you cannot stop it.'
      return false
    end
  end
end
```

```
end

ec2_client.stop_instances(instance_ids: [instance_id])
ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
puts 'Instance stopped.'
true
rescue StandardError => e
  puts "Error stopping instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-stop-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to stop instance '#{instance_id}' " \
    '(this might take a few minutes)... '
  return if instance_stopped?(ec2_client, instance_id)

  puts 'Could not stop instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- API-Einheiten finden Sie [StopInstances](#) in der AWS SDK for Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error>
{
    tracing::info!("Stopping instance {instance_id}");

    self.client
        .stop_instances()
        .instance_ids(instance_id)
        .send()
        .await?;

    self.wait_for_instance_stopped(instance_id, None).await?;

    tracing::info!("Stopped instance.");

    Ok(())
}
```

Warten Sie, bis sich eine Instanz im gestoppten Zustand befindet, und verwenden Sie die Waiters. API Die Verwendung von Waiters API erfordert `use aws_sdk_ec2: :client: :Waiters`` in der Rust-Datei.

```
pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error>
{
    tracing::info!("Stopping instance {instance_id}");

    self.client
```

```

        .stop_instances()
        .instance_ids(instance_id)
        .send()
        .await?;

self.wait_for_instance_stopped(instance_id, None).await?;

tracing::info!("Stopped instance.");

Ok(())
}

```

- Einzelheiten [StopInstances](#) finden AWS SDK Sie in der Rust-Referenz. API API

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

DATA lt_instance_ids TYPE /aws1/
cl_ec2instidstringlist_w=>tt_instanceidstringlist.
APPEND NEW /aws1/cl_ec2instidstringlist_w( iv_value = iv_instance_id ) TO
lt_instance_ids.

"Perform dry run"
TRY.
    " DryRun is set to true. This checks for the required permissions to stop
the instance without actually making the request. "
    lo_ec2->stopinstances(
        it_instanceids = lt_instance_ids
        iv_dryrun = abap_true
    ).
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    " If the error code returned is `DryRunOperation`, then you have the
required permissions to stop this instance. "

```

```

    IF lo_exception->av_err_code = 'DryRunOperation'.
        MESSAGE 'Dry run to stop instance completed.' TYPE 'I'.
        " DryRun is set to false to stop instance. "
        oo_result = lo_ec2->stopinstances(          " oo_result is returned
for testing purposes. "
            it_instanceids = lt_instance_ids
            iv_dryrun = abap_false
        ).
        MESSAGE 'Successfully stopped the EC2 instance.' TYPE 'I'.
        " If the error code returned is `UnauthorizedOperation`, then you don't
have the required permissions to stop this instance. "
        ELSEIF lo_exception->av_err_code = 'UnauthorizedOperation'.
            MESSAGE 'Dry run to stop instance failed. User does not have
permissions to stop the instance.' TYPE 'E'.
        ELSE.
            DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
            MESSAGE lv_error TYPE 'E'.
        ENDIF.
    ENDTRY.

```

- API-Einheiten finden Sie [StopInstances](#) unter AWS SDKSAPABAPI als Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **TerminateInstances** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `TerminateInstances`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/// <summary>
/// Terminate an EC2 instance.
/// </summary>
/// <param name="ec2InstanceId">The instance Id of the EC2 instance
/// to terminate.</param>
/// <returns>Async task.</returns>
public async Task<List<InstanceStateChange>> TerminateInstances(string
ec2InstanceId)
{
    try
    {
        var request = new TerminateInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId }
        };

        var response = await _amazonEC2.TerminateInstancesAsync(request);
        Console.WriteLine("Waiting for the instance to terminate.");
        await WaitForInstanceState(ec2InstanceId,
InstanceStateName.Terminated);

        Console.WriteLine($"\\nThe instance {ec2InstanceId} has been
terminated.");
        return response.TerminatingInstances;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidInstanceId")
        {
            _logger.LogError(
                $"InstanceId is invalid, unable to terminate.
{ec2Exception.Message}");
        }
    }
}
```



```
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(
            $"An error occurred while terminating the instance.:
{ex.Message}");
        throw;
    }
}

/// <summary>
/// Wait until an EC2 instance is in a specified state.
/// </summary>
/// <param name="instanceId">The instance Id.</param>
/// <param name="stateName">The state to wait for.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
{
    var request = new DescribeInstancesRequest
    {
        InstanceIds = new List<string> { instanceId }
    };

    // Wait until the instance is in the specified state.
    var hasState = false;
    do
    {
        // Wait 5 seconds.
        Thread.Sleep(5000);

        // Check for the desired state.
        var response = await _amazonEC2.DescribeInstancesAsync(request);
        var instance = response.Reservations[0].Instances[0];
        hasState = instance.State.Name == stateName;
        Console.WriteLine(". ");
    } while (!hasState);

    return hasState;
}
```

- API-Einheiten finden Sie [TerminateInstances](#) in der AWS SDK for .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#####
# function ec2_terminate_instances
#
# This function terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
# instances using the AWS CLI.
#
# Parameters:
#     -i instance_ids - A space-separated list of instance IDs.
#     -h - Display help.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_terminate_instances() {
    local instance_ids response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_terminate_instances"
        echo "Terminates one or more Amazon Elastic Compute Cloud (Amazon EC2)
instances."
        echo "  -i instance_ids - A space-separated list of instance IDs."
        echo "  -h - Display help."
        echo ""
    }
}
```

```

# Retrieve the calling parameters.
while getopts "i:h" option; do
  case "${option}" in
    i) instance_ids="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

# Check if instance ID is provided
if [[ -z "${instance_ids}" ]]; then
  echo "Error: Missing required instance IDs parameter."
  usage
  return 1
fi

# shellcheck disable=SC2086
response=$(aws ec2 terminate-instances \
  "--instance-ids" $instance_ids \
  "--query 'TerminatingInstances[*].[InstanceId,CurrentState.Name]' \
  "--output text) || {
  aws_cli_error_log ${?}
  errecho "ERROR: AWS reports terminate-instances operation failed.$response"
  return 1
}

return 0
}

```

Die in diesem Beispiel verwendeten Dienstprogrammfunktionen.

```

#####
# function errecho
#

```

```

# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function aws_cli_error_log()
#
# This function is used to log the error messages from the AWS CLI.
#
# The function expects the following argument:
#     $1 - The error code returned by the AWS CLI.
#
# Returns:
#     0: - Success.
#
#####
function aws_cli_error_log() {
    local err_code=$1
    errecho "Error code : $err_code"
    if [ "$err_code" == 1 ]; then
        errecho " One or more S3 transfers failed."
    elif [ "$err_code" == 2 ]; then
        errecho " Command line failed to parse."
    elif [ "$err_code" == 130 ]; then
        errecho " Process received SIGINT."
    elif [ "$err_code" == 252 ]; then
        errecho " Command syntax invalid."
    elif [ "$err_code" == 253 ]; then
        errecho " The system environment or configuration was invalid."
    elif [ "$err_code" == 254 ]; then
        errecho " The service returned an error."
    elif [ "$err_code" == 255 ]; then
        errecho " 255 is a catch-all error."
    fi

    return 0
}

```

- API-Einheiten finden Sie [TerminateInstances](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
#!/ Terminate an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::terminateInstances(const Aws::String &instanceID,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::TerminateInstancesRequest request;
    request.SetInstanceIds({instanceID});

    Aws::EC2::Model::TerminateInstancesOutcome outcome =
        ec2Client.TerminateInstances(request);
    if (outcome.IsSuccess()) {
        std::cout << "Ec2 instance '" << instanceID <<
            "' was terminated." << std::endl;
    } else {
        std::cerr << "Failed to terminate ec2 instance '" << instanceID <<
            "', " <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    return outcome.IsSuccess();
}
```

- API Einzelheiten finden Sie [TerminateInstances](#) in der AWS SDK for C++ API Referenz.

CLI

AWS CLI

Um eine EC2 Amazon-Instance zu beenden

In diesem Beispiel wird die angegebene Instance beendet.

Befehl:

```
aws ec2 terminate-instances --instance-ids i-1234567890abcdef0
```

Ausgabe:


```
{
  "TerminatingInstances": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "CurrentState": {
        "Code": 32,
        "Name": "shutting-down"
      },
      "PreviousState": {
        "Code": 16,
        "Name": "running"
      }
    }
  ]
}
```

Weitere Informationen finden Sie unter [Using Amazon EC2 Instances](#) im AWS Command Line Interface User Guide.

- API-Einheiten finden Sie [TerminateInstances](#) unter AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
/**
 * Terminates an EC2 instance asynchronously and waits for it to reach the
 * terminated state.
 *
 * @param instanceId the ID of the EC2 instance to terminate
 * @return a {@link CompletableFuture} that completes when the instance has
 * been terminated
 * @throws RuntimeException if there is no response from the AWS SDK or if
 * there is a failure during the termination process
 */
public CompletableFuture<Object> terminateEC2Async(String instanceId) {
    TerminateInstancesRequest terminateRequest =
    TerminateInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    CompletableFuture<TerminateInstancesResponse> responseFuture =
    getAsyncClient().terminateInstances(terminateRequest);
    return responseFuture.thenCompose(terminateResponse -> {
        if (terminateResponse == null) {
            throw new RuntimeException("No response received for terminating
            instance " + instanceId);
        }
        System.out.println("Going to terminate an EC2 instance and use a
        waiter to wait for it to be in terminated state");
        return getAsyncClient().waiter()
            .waitUntilInstanceTerminated(r -> r.instanceIds(instanceId))
            .thenApply(waiterResponse -> null);
    }).exceptionally(throwable -> {
        // Handle any exceptions that occurred during the async call
        throw new RuntimeException("Failed to terminate EC2 instance: " +
        throwable.getMessage(), throwable);
    });
}
```

```
    });
  }
```

- API-Einheiten finden Sie [TerminateInstances](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import { EC2Client, TerminateInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Terminate one or more EC2 instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new TerminateInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { TerminatingInstances } = await client.send(command);
    const instanceList = TerminatingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Terminating instances:");
    console.log(instanceList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
```



```
        console.warn(`${caught.message}`);
    } else {
        throw caught;
    }
}
};
```

- API-Einheiten finden Sie [TerminateInstances](#) in der AWS SDK for JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
suspend fun terminateEC2(instanceID: String) {
    val request =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceID)
        }

    Ec2Client { region = "us-west-2" }.use { ec2 ->
        val response = ec2.terminateInstances(request)
        response.terminatingInstances?.forEach { instance ->
            println("The ID of the terminated instance is
                ${instance.instanceId}")
        }
    }
}
```

- API-Einheiten finden Sie [TerminateInstances](#) in der AWS SDK API Kotlin-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die angegebene Instanz beendet (die Instanz läuft möglicherweise oder befindet sich im Status „gestoppt“). Das Cmdlet fordert Sie zur Bestätigung auf, bevor Sie fortfahren. Verwenden Sie die Befehlszeilenoption `-Force`, um die Aufforderung zu unterdrücken.

```
Remove-EC2Instance -InstanceId i-12345678
```

Ausgabe:

CurrentState	InstanceId	PreviousState
-----	-----	-----
Amazon.EC2.Model.InstanceState	i-12345678	Amazon.EC2.Model.InstanceState

- API-Einheiten finden Sie unter [TerminateInstances](#) Cmdlet-Referenz. AWS Tools for PowerShell

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.
```

```
        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
        access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
objects. These are high-level objects that
        wrap instance actions.
    """
    self.ec2_client = ec2_client
    self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def terminate(self) -> None:
        """
        Terminates instances and waits for them to reach the terminated state.
        """
        if not self.instances:
            logger.info("No instances to terminate.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            self.ec2_client.terminate_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_terminated")
            waiter.wait(InstanceIds=instance_ids)
            self.instances.clear()
            for instance_id in instance_ids:
                print(f"• Instance ID: {instance_id}\n" f"• Action: Terminated")

        except ClientError as err:
            logger.error(
                f"Failed instance termination details:\n\t{str(self.instances)}"
            )
```

```
error_code = err.response["Error"]["Code"]
if error_code == "InvalidInstanceID.NotFound":
    logger.error(
        "One or more instance IDs do not exist. "
        "Please verify the instance IDs and try again."
    )
raise
```

- API-Einheiten finden Sie unter [TerminateInstances](#) Python (Boto3) API -Referenz. AWS SDK

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
require 'aws-sdk-ec2'

# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was terminated; otherwise, false.
# @example
#   exit 1 unless instance_terminated?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'i-123abc'
#   )
def instance_terminated?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])
```

```
if response.instance_statuses.count.positive? &&
  response.instance_statuses[0].instance_state.name == 'terminated'

  puts 'The instance is already terminated.'
  return true
end

ec2_client.terminate_instances(instance_ids: [instance_id])
ec2_client.wait_until(:instance_terminated, instance_ids: [instance_id])
puts 'Instance terminated.'
true
rescue StandardError => e
  puts "Error terminating instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to terminate instance '#{instance_id}' " \
    '(this might take a few minutes)... '
  return if instance_terminated?(ec2_client, instance_id)
end
```

```

    puts 'Could not terminate instance.'
  end

  run_me if $PROGRAM_NAME == __FILE__

```

- API-Einheiten finden Sie [TerminateInstances](#) in der AWS SDK for Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```

pub async fn delete_instance(&self, instance_id: &str) -> Result<(),
EC2Error> {
    tracing::info!("Deleting instance with id {instance_id}");
    self.stop_instance(instance_id).await?;
    self.client
        .terminate_instances()
        .instance_ids(instance_id)
        .send()
        .await?;
    self.wait_for_instance_terminated(instance_id).await?;
    tracing::info!("Terminated instance with id {instance_id}");
    Ok(())
}

```

Warten Sie, bis sich eine Instanz im Status „Beendet“ befindet, indem Sie die API Waiters verwenden. Die Verwendung von Waiters API erfordert `use aws_sdk_ec2: :client: :Waiters` in der Rust-Datei.

```

    async fn wait_for_instance_terminated(&self, instance_id: &str) -> Result<(),
EC2Error> {
        self.client

```

```

        .wait_until_instance_terminated()
        .instance_ids(instance_id)
        .wait(Duration::from_secs(60))
        .await
        .map_err(|err| match err {
            WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                "Exceeded max time ({}s) waiting for instance to terminate.",
                exceeded.max_wait().as_secs(),
            )),
            _ => EC2Error::from(err),
        })?;
    Ok(())
}

```

- Einzelheiten [TerminateInstances](#) finden Sie in der Rust-Referenz. API API

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Verwenden Sie **UnassignPrivateIpAddresses** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `UnassignPrivateIpAddresses`.

CLI

AWS CLI

Um die Zuweisung einer sekundären privaten IP-Adresse zu einer Netzwerkschnittstelle aufzuheben

In diesem Beispiel wird die Zuweisung der angegebenen privaten IP-Adresse zur angegebenen Netzwerkschnittstelle aufgehoben. Wird der Befehl erfolgreich ausgeführt, wird keine Ausgabe zurückgegeben.

Befehl:

```
aws ec2 unassign-private-ip-addresses --network-interface-id eni-e5aa89a3 --private-ip-addresses 10.0.0.82
```

- API Einzelheiten finden Sie unter [UnassignPrivateIpAddresses AWS CLI](#) Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die Zuweisung der angegebenen privaten IP-Adresse zur angegebenen Netzwerkschnittstelle aufgehoben.

```
Unregister-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress 10.0.0.82
```

- API-Einheiten finden Sie unter [UnassignPrivateIpAddresses AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **UnmonitorInstances** mit einem AWS SDK oder CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `UnmonitorInstances`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erlernen Sie die Grundlagen](#)

C++

SDK für C++

Note

Es gibt noch mehr dazu [GitHub](#). Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
//! Disable monitoring for an EC2 instance.  
/*!  
    \param instanceId: An EC2 instance ID.  
    \param clientConfiguration: AWS client configuration.
```



```

    \return bool: Function succeeded.
    */
bool AwsDoc::EC2::disableMonitoring(const Aws::String &instanceId,
                                   const Aws::Client::ClientConfiguration
                                   &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::UnmonitorInstancesRequest unrequest;
    unrequest.AddInstanceIds(instanceId);
    unrequest.SetDryRun(true);

    Aws::EC2::Model::UnmonitorInstancesOutcome dryRunOutcome =
ec2Client.UnmonitorInstances(unrequest);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to disable monitoring on instance. A dry run
should trigger an error."
            <<
            std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
                Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to disable monitoring on instance " <<
            instanceId << ": " << dryRunOutcome.GetError().GetMessage() <<
            std::endl;
        return false;
    }

    unrequest.SetDryRun(false);
    Aws::EC2::Model::UnmonitorInstancesOutcome unmonitorInstancesOutcome =
ec2Client.UnmonitorInstances(unrequest);
    if (!unmonitorInstancesOutcome.IsSuccess()) {
        std::cout << "Failed to disable monitoring on instance " << instanceId
            << ": " << unmonitorInstancesOutcome.GetError().GetMessage() <<
            std::endl;
    } else {
        std::cout << "Successfully disable monitoring on instance " <<
            instanceId << std::endl;
    }

    return unmonitorInstancesOutcome.IsSuccess();
}

```

- API-Einheiten finden Sie [UnmonitorInstances](#) in der AWS SDK for C++ API-Referenz.

CLI

AWS CLI

So deaktivieren Sie die detaillierte Überwachung für eine Instance

Dieser Beispielbefehl deaktiviert die detaillierte Überwachung für die angegebene Instance.

Befehl:

```
aws ec2 unmonitor-instances --instance-ids i-1234567890abcdef0
```

Ausgabe:

```
{
  "InstanceMonitorings": [
    {
      "InstanceId": "i-1234567890abcdef0",
      "Monitoring": {
        "State": "disabling"
      }
    }
  ]
}
```

- API-Einheiten finden Sie [UnmonitorInstances](#) unter AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

```
import { EC2Client, UnmonitorInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";
```

```

/**
 * Turn off detailed monitoring for the selected instance.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new UnmonitorInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { InstanceMonitorings } = await client.send(command);
    const instanceMonitoringsList = InstanceMonitorings.map(
      (im) =>
        ` • Detailed monitoring state for ${im.InstanceId} is
        ${im.Monitoring.State}.`,
    );
    console.log("Monitoring status:");
    console.log(instanceMonitoringsList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(` ${caught.message} `);
    } else {
      throw caught;
    }
  }
};

```

- API-Einheiten finden Sie [UnmonitorInstances](#) in der AWS SDK for JavaScript API-Referenz.

PowerShell

Tools für PowerShell

Beispiel 1: In diesem Beispiel wird die detaillierte Überwachung für die angegebene Instanz deaktiviert.

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

Ausgabe:

```
InstanceId      Monitoring
-----
i-12345678     Amazon.EC2.Model.Monitoring
```

- API-Einheiten finden Sie unter [UnmonitorInstances AWS Tools for PowerShell](#) Cmdlet-Referenz.

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **UpdateSecurityGroupRuleDescriptionsIngress** mit einem CLI

Die folgenden Codebeispiele zeigen, wie man es benutzt `UpdateSecurityGroupRuleDescriptionsIngress`.

CLI

AWS CLI

Beispiel 1: Um die Beschreibung einer Sicherheitsgruppenregel für eingehenden Datenverkehr mit einer CIDR-Quelle zu aktualisieren

Im folgenden `update-security-group-rule-descriptions-ingress` Beispiel wird die Beschreibung der Sicherheitsgruppenregel für den angegebenen Port und IPv4-Adressbereich aktualisiert. Die Beschreibung `'SSH access from ABC office'` ersetzt alle vorhandenen Beschreibungen für die Regel.

```
aws ec2 update-security-group-rule-descriptions-ingress \
  --group-id sg-02f0d35a850ba727f \
  --ip-permissions
  IpProtocol=tcp,FromPort=22,ToPort=22,IpRanges='[{"CidrIp=203.0.113.0/16,Description="SSH
  access from corpnet"}]'
```

Ausgabe:

```
{
```

```
"Return": true
}
```

Weitere Informationen finden Sie unter [Regeln für Sicherheitsgruppen](#) im EC2Amazon-Benutzerhandbuch.

Beispiel 2: Um die Beschreibung einer Sicherheitsgruppenregel für eingehenden Datenverkehr mit einer Präfixliste zu aktualisieren

Im folgenden `update-security-group-rule-descriptions-ingress` Beispiel wird die Beschreibung der Sicherheitsgruppenregel für die angegebene Port- und Präfixliste aktualisiert. Die Beschreibung 'SSH access from ABC office' ersetzt alle vorhandenen Beschreibungen für die Regel.

```
aws ec2 update-security-group-rule-descriptions-ingress \
  --group-id sg-02f0d35a850ba727f \
  --ip-permissions
  IpProtocol=tcp,FromPort=22,ToPort=22,PrefixListIds='[{PrefixListId=pl-12345678,Description=SSH
  access from corpnet}]'
```

Ausgabe:

```
{
  "Return": true
}
```

Weitere Informationen finden Sie unter [Regeln für Sicherheitsgruppen](#) im EC2Amazon-Benutzerhandbuch.

- API-Einheiten finden Sie [UpdateSecurityGroupRuleDescriptionsIngress](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell

Beispiel 1: Aktualisiert die Beschreibung einer bestehenden Sicherheitsgruppenregel für eingehenden Datenverkehr.

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupId
"sg-1234567890"
```

```
$ruleWithUpdatedDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
    "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
    "Description" = "Updated rule description"
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId
$existingInboundRule.GroupId -SecurityGroupRuleDescription
$ruleWithUpdatedDescription
```

Beispiel 2: Entfernt die Beschreibung einer vorhandenen Sicherheitsgruppenregel für eingehenden (eingehenden) Datenverkehr (indem der Parameter in der Anforderung weggelassen wird).

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupRuleId
"sgr-1234567890"
$ruleWithoutDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
    "SecurityGroupRuleId" = $existingInboundRule.SecurityGroupRuleId
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId
$existingInboundRule.GroupId -SecurityGroupRuleDescription
$ruleWithoutDescription
```

- API-Einheiten finden Sie unter [UpdateSecurityGroupRuleDescriptionsIngress](#) Cmdlet-Referenz. AWS Tools for PowerShell

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Szenarien für die EC2-Verwendung von Amazon AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon EC2 mit implementierten AWS SDKs. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben erledigen können, indem Sie mehrere Funktionen innerhalb von Amazon EC2 oder in Kombination mit anderen aufrufen AWS-Services. Jedes Szenario enthält einen Link zum vollständigen Quellcode, in dem Sie Anweisungen zur Einrichtung und Ausführung des Codes finden.

Die Szenarien zielen auf ein mittleres Erfahrungsniveau ab, damit Sie Serviceaktionen im Kontext besser verstehen können.

Beispiele

- [Erstellen und verwalten Sie einen ausfallsicheren Service mithilfe eines AWS SDK](#)

Erstellen und verwalten Sie einen ausfallsicheren Service mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie einen Webservice mit Lastenausgleich erstellen, der Buch-, Film- und Liedempfehlungen zurückgibt. Das Beispiel zeigt, wie der Service auf Fehler reagiert und wie der Service für mehr Ausfallsicherheit umstrukturiert werden kann.

- Verwenden Sie eine Amazon EC2 Auto Scaling Scaling-Gruppe, um Amazon Elastic Compute Cloud (AmazonEC2) -Instances auf der Grundlage einer Startvorlage zu erstellen und die Anzahl der Instances in einem bestimmten Bereich zu halten.
- Bearbeiten und verteilen Sie HTTP Anfragen mit Elastic Load Balancing.
- Überwachen Sie den Zustand von Instances in einer Auto-Scaling-Gruppe und leiten Sie Anfragen nur an fehlerfreie Instances weiter.
- Führen Sie auf jeder EC2 Instanz einen Python-Webserver aus, um HTTP Anfragen zu bearbeiten. Der Webserver reagiert mit Empfehlungen und Zustandsprüfungen.
- Simulieren Sie einen Empfehlungsservice mit einer Amazon DynamoDB-Tabelle.
- Steuern Sie die Antwort des Webserver auf Anfragen und Zustandsprüfungen, indem Sie die AWS Systems Manager Parameter aktualisieren.

.NET

AWS SDK for .NET

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
static async Task Main(string[] args)
{
    _configuration = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("settings.json") // Load settings from .json file.
        .AddJsonFile("settings.local.json",
            true) // Optionally, load local settings.
        .Build();

    // Set up dependency injection for the AWS services.
    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<IAmazonIdentityManagementService>()
                .AddAWSService<IAmazonDynamoDB>()
                .AddAWSService<IAmazonElasticLoadBalancingV2>()
                .AddAWSService<IAmazonSimpleSystemsManagement>()
                .AddAWSService<IAmazonAutoScaling>()
                .AddAWSService<IAmazonEC2>()
                .AddTransient<AutoScalerWrapper>()
                .AddTransient<ElasticLoadBalancerWrapper>()
                .AddTransient<SmParameterWrapper>()
                .AddTransient<Recommendations>()
                .AddSingleton<IConfiguration>(_configuration)
            )
        .Build();

    ServicesSetup(host);
    ResourcesSetup();

    try
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
        Console.WriteLine(new string('-', 80));
        await Deploy(true);
    }
}
```



```
        Console.WriteLine("Now let's begin the scenario.");
        Console.WriteLine(new string('-', 80));
        await Demo(true);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Finally, let's clean up our resources.");
        Console.WriteLine(new string('-', 80));

        await DestroyResources(true);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
        Console.WriteLine(new string('-', 80));
    }
    catch (Exception ex)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
        await DestroyResources(true);
        Console.WriteLine(new string('-', 80));
    }
}

/// <summary>
/// Setup any common resources, also used for integration testing.
/// </summary>
public static void ResourcesSetup()
{
    _httpClient = new HttpClient();
}

/// <summary>
/// Populate the services for use within the console application.
/// </summary>
/// <param name="host">The services host.</param>
private static void ServicesSetup(IHost host)
{
    _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
    _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
```

```
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper =
host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <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> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
            "\nFor this demo, we'll use the AWS SDK for .NET to create several
AWS resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server.");
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();
    }
}
```

```
// Create and populate the DynamoDB table.
var databaseTableName = _configuration["databaseName"];
var recommendationsPath = Path.Join(_configuration["resourcePath"],
    "recommendations_objects.json");
Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
await _recommendations.CreateDatabaseWithName(databaseTableName);
await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
Console.WriteLine(new string('-', 80));

// Create the EC2 Launch Template.

Console.WriteLine(
    $"Creating an EC2 launch template that runs
'server_startup_script.sh' when an instance starts.\n"
    + "\nThis script starts a Python web server defined in the
'server.py' script. The web server\n"
    + "listens to HTTP requests on port 80 and responds to requests to
 '/' and to '/healthcheck'.\n"
    + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
    + "run a web server, such as Apache, with least-privileged
credentials.");
Console.WriteLine(
    "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
    + "permissions to access the DynamoDB recommendation table and
Systems Manager parameters\n"
    + "that control the flow of the demo.");

var startupScriptPath = Path.Join(_configuration["resourcePath"],
    "server_startup_script.sh");
var instancePolicyPath = Path.Join(_configuration["resourcePath"],
    "instance_policy.json");
await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
Console.WriteLine(new string('-', 80));

Console.WriteLine(
    "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
    + "Availability Zone.\n");
var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
```

```
        await _autoScalerWrapper.CreateGroupOfSize(3,
_autoScalerWrapper.GroupName, zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or
continue with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the
demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
The target group\n"
            + "defines how the load balancer connects to instances. The load
balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
_autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
        var targetGroup = await
_elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGro
protocol, port, defaultVpc.VpcId);

        await
_elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.Lo
subnetIds, targetGroup);
        await
_autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
_elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper
```

```
        var loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
_autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
_autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port,
ipString);
            var sshPortIsOpen =
_autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
ipString);

            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for
your default VPC must\n"
                    + "allows access from this computer. You can either add it
automatically from this\n"
                    + "example or add it yourself using the AWS Management
Console.\n");

                if (!interactive || GetYesNoResponse(
                    "Do you want to add a rule to the security group to allow
inbound traffic from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port,
ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
```

```
        "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
        {
            await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
ipString);
        }
    }
    loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
}

if (loadBalancerAccess)
{
    Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
    Console.WriteLine($"\\thttp://{endPoint}\\n");
}
else
{
    Console.WriteLine(
        "\\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\\n"
        + "manually verifying that your VPC and security group are
configured correctly and that\\n"
        + "you can successfully make a GET request to the load balancer
endpoint:\\n");
    Console.WriteLine($"\\thttp://{endPoint}\\n");
}
Console.WriteLine(new string('-', 80));
Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
if (interactive)
    Console.ReadLine();
return true;
}

/// <summary>
/// Demonstrate the steps of the scenario.
/// </summary>
/// <param name="interactive">True to run as an interactive scenario.</param>
/// <returns>Async task.</returns>
public static async Task<bool> Demo(bool interactive)
{
```

```
var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
    "ssm_only_policy.json");

Console.WriteLine(new string('-', 80));
Console.WriteLine("Resetting parameters to starting values for demo.");
await _smParameterWrapper.Reset();

Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
    "to create situations where the web service fails, and
shows how using a resilient\n" +
    "architecture can keep the web service running in spite
of these failures.");
Console.WriteLine(new string('-', 88));
Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
if (interactive)
    await DemoActionChoices();

Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
    $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
    $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
    "this-is-not-a-table");
Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
    "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
if (interactive)
    await DemoActionChoices();

Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
    "static");
```

```
        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
            $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
```



```
        await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"Now, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"Even while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
```

```
        Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");

        Console.WriteLine($"When all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n" +
            "that were created for this demo."
        );
    }
}
```

```

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
            _elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
            _elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
            _autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
            _autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
            _autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
            _recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        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;
    }

```

Erstellen Sie eine Klasse, die Auto Scaling- und EC2 Amazon-Aktionen umschließt.

```

/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;

```

```
private readonly IAmazonEC2 _amazonEc2;
private readonly IAmazonSimpleSystemsManagement _amazonSsm;
private readonly IAmazonIdentityManagementService _amazonIam;
private readonly ILogger<AutoScalerWrapper> _logger;

private readonly string _instanceType = "";
private readonly string _amiParam = "";
private readonly string _launchTemplateName = "";
private readonly string _groupName = "";
private readonly string _instancePolicyName = "";
private readonly string _instanceRoleName = "";
private readonly string _instanceProfileName = "";
private readonly string _badCredsProfileName = "";
private readonly string _badCredsRoleName = "";
private readonly string _badCredsPolicyName = "";
private readonly string _keyPairName = "";

public string GroupName => _groupName;
public string KeyPairName => _keyPairName;
public string LaunchTemplateName => _launchTemplateName;
public string InstancePolicyName => _instancePolicyName;
public string BadCredsProfileName => _badCredsProfileName;
public string BadCredsRoleName => _badCredsRoleName;
public string BadCredsPolicyName => _badCredsPolicyName;

/// <summary>
/// Constructor for the AutoScalerWrapper.
/// </summary>
/// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
/// <param name="amazonEc2">The injected EC2 client.</param>
/// <param name="amazonIam">The injected IAM client.</param>
/// <param name="amazonSsm">The injected SSM client.</param>
public AutoScalerWrapper(
    IAmazonAutoScaling amazonAutoScaling,
    IAmazonEC2 amazonEc2,
    IAmazonSimpleSystemsManagement amazonSsm,
    IAmazonIdentityManagementService amazonIam,
    IConfiguration configuration,
    ILogger<AutoScalerWrapper> logger)
{
    _amazonAutoScaling = amazonAutoScaling;
    _amazonEc2 = amazonEc2;
    _amazonSsm = amazonSsm;
    _amazonIam = amazonIam;
}
```

```

    _logger = logger;

    var prefix = configuration["resourcePrefix"];
    _instanceType = configuration["instanceType"];
    _amiParam = configuration["amiParam"];

    _launchTemplateName = prefix + "-template";
    _groupName = prefix + "-group";
    _instancePolicyName = prefix + "-pol";
    _instanceRoleName = prefix + "-role";
    _instanceProfileName = prefix + "-prof";
    _badCredsPolicyName = prefix + "-bc-pol";
    _badCredsRoleName = prefix + "-bc-role";
    _badCredsProfileName = prefix + "-bc-prof";
    _keyPairName = prefix + "-key-pair";
}

/// <summary>
/// Create a policy, role, and profile that is associated with instances with
a specified name.
/// An instance's associated profile defines a role that is assumed by the
/// instance. The role has attached policies that specify the AWS permissions
granted to
/// clients that run on the instance.
/// </summary>
/// <param name="policyName">Name to use for the policy.</param>
/// <param name="roleName">Name to use for the role.</param>
/// <param name="profileName">Name to use for the profile.</param>
/// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
/// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
/// <returns>The Arn of the profile.</returns>
public async Task<string> CreateInstanceProfileWithName(
    string policyName,
    string roleName,
    string profileName,
    string ssmOnlyPolicyFile,
    List<string>? awsManagedPolicies = null)
{

    var assumeRoleDoc = "{" +
                                "\"Version\": \"2012-10-17\", " +
                                "\"Statement\": [{" +
                                    "\"Effect\": \"Allow\", " +

```

```
        "\"Principal\": {" +
        "\"Service\": [" +
            "\"ec2.amazonaws.com\"" +
        "]" +
        "}," +
        "\"Action\": \"sts:AssumeRole\"" +
        "]" +
    "}";

var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

var policyArn = "";

try
{
    var createPolicyResult = await _amazonIam.CreatePolicyAsync(
        new CreatePolicyRequest
        {
            PolicyName = policyName,
            PolicyDocument = policyDocument
        });
    policyArn = createPolicyResult.Policy.Arn;
}
catch (EntityAlreadyExistsException)
{
    // The policy already exists, so we look it up to get the Arn.
    var policiesPaginator = _amazonIam.Paginators.ListPolicies(
        new ListPoliciesRequest()
        {
            Scope = PolicyScopeType.Local
        });
    // Get the entire list using the paginator.
    await foreach (var policy in policiesPaginator.Policies)
    {
        if (policy.PolicyName.Equals(policyName))
        {
            policyArn = policy.Arn;
        }
    }

    if (policyArn == null)
    {
        throw new InvalidOperationException("Policy not found");
    }
}
```

```
    }

    try
    {
        await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = assumeRoleDoc,
        });
        await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
        {
            RoleName = roleName,
            PolicyArn = policyArn
        });
        if (awsManagedPolicies != null)
        {
            foreach (var awsPolicy in awsManagedPolicies)
            {
                await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                {
                    PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                    RoleName = roleName
                });
            }
        }
    }
    catch (EntityAlreadyExistsException)
    {
        Console.WriteLine("Role already exists.");
    }

    string profileArn = "";
    try
    {
        var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
        new CreateInstanceProfileRequest()
        {
            InstanceProfileName = profileName
        });
        // Allow time for the profile to be ready.
        profileArn = profileCreateResponse.InstanceProfile.Arn;
        Thread.Sleep(10000);
    }
}
```

```
        await _amazonIam.AddRoleToInstanceProfileAsync(
            new AddRoleToInstanceProfileRequest()
            {
                InstanceProfileName = profileName,
                RoleName = roleName
            });
    }
    catch (EntityAlreadyExistsException)
    {
        Console.WriteLine("Policy already exists.");
        var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
            new GetInstanceProfileRequest()
            {
                InstanceProfileName = profileName
            });
        profileArn = profileGetResponse.InstanceProfile.Arn;
    }
    return profileArn;
}

/// <summary>
/// Create a new key pair and save the file.
/// </summary>
/// <param name="newKeyPairName">The name of the new key pair.</param>
/// <returns>Async task.</returns>
public async Task CreateKeyPair(string newKeyPairName)
{
    try
    {
        var keyResponse = await _amazonEc2.CreateKeyPairAsync(
            new CreateKeyPairRequest() { KeyName = newKeyPairName });
        await File.WriteAllTextAsync($"{newKeyPairName}.pem",
            keyResponse.KeyPair.KeyMaterial);
        Console.WriteLine($"Created key pair {newKeyPairName}.");
    }
    catch (AlreadyExistsException)
    {
        Console.WriteLine("Key pair already exists.");
    }
}

/// <summary>
/// Delete the key pair and file by name.
```



```
/// </summary>
/// <param name="deleteKeyPairName">The key pair to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteKeyPairByName(string deleteKeyPairName)
{
    try
    {
        await _amazonEc2.DeleteKeyPairAsync(
            new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
        File.Delete($"{deleteKeyPairName}.pem");
    }
    catch (FileNotFoundException)
    {
        Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
    }
}

/// <summary>
/// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
/// The launch template specifies a Bash script in its user data field that
runs after
/// the instance is started. This script installs the Python packages and
starts a Python
/// web server on the instance.
/// </summary>
/// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
/// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
/// <returns>The template object.</returns>
public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
{
    try
    {
        await CreateKeyPair(_keyPairName);
        await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
_instanceProfileName, instancePolicyPath);

        var startServerText = await File.ReadAllTextAsync(startupScriptPath);
        var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);
```

```
var amiLatest = await _amazonSsm.GetParameterAsync(
    new GetParameterRequest() { Name = _amiParam });
var amiId = amiLatest.Parameter.Value;
var launchTemplateResponse = await
_amazonEc2.CreateLaunchTemplateAsync(
    new CreateLaunchTemplateRequest()
    {
        LaunchTemplateName = _launchTemplateName,
        LaunchTemplateData = new RequestLaunchTemplateData()
        {
            InstanceType = _instanceType,
            ImageId = amiId,
            IamInstanceProfile =
                new
LaunchTemplateIamInstanceProfileSpecificationRequest()
                {
                    Name = _instanceProfileName
                },
            KeyName = _keyPairName,
            UserData = System.Convert.ToBase64String(plainTextBytes)
        }
    });
return launchTemplateResponse.LaunchTemplate;
}
catch (AmazonEC2Exception ec2Exception)
{
    if (ec2Exception.ErrorCode ==
        "InvalidLaunchTemplateName.AlreadyExistsException")
    {
        _logger.LogError($"Could not create the template, the name
        {_launchTemplateName} already exists. " +
            $"Please try again with a unique name.");
    }

    throw;
}
catch (Exception ex)
{
    _logger.LogError($"An error occurred while creating the template.:
    {ex.Message}");
    throw;
}
```

```
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2
    Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z =>
z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability
zones.: {ex.Message}");
            throw;
        }
    }
}

/// <summary>
/// Create an EC2 Auto Scaling group of a specified size and name.
/// </summary>
/// <param name="groupSize">The size for the group.</param>
/// <param name="groupName">The name for the group.</param>
/// <param name="availabilityZones">The availability zones for the group.</
param>
/// <returns>Async task.</returns>
public async Task CreateGroupOfSize(int groupSize, string groupName,
List<string> availabilityZones)
{
    try
    {
        await _amazonAutoScaling.CreateAutoScalingGroupAsync(
```

```
        new CreateAutoScalingGroupRequest()
        {
            AutoScalingGroupName = groupName,
            AvailabilityZones = availabilityZones,
            LaunchTemplate =
                new
Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                {
                    LaunchTemplateName = _launchTemplateName,
                    Version = "$Default"
                },
            MaxSize = groupSize,
            MinSize = groupSize
        });
        Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
size {groupSize}.");
    }
    catch (EntityAlreadyExistsException)
    {
        Console.WriteLine($"EC2 Auto Scaling group {groupName} already
exists.");
    }
}

/// <summary>
/// Get the default VPC for the account.
/// </summary>
/// <returns>The default VPC object.</returns>
public async Task<Vpc> GetDefaultVpc()
{
    try
    {
        var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
            new DescribeVpcsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new("is-default", new List<string>() { "true" })
                }
            });
        return vpcResponse.Vpcs[0];
    }
    catch (AmazonEC2Exception ec2Exception)
    {
```

```
        if (ec2Exception.ErrorCode == "UnauthorizedOperation")
        {
            _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"An error occurred while describing the vpcs.:
{ex.Message}");
        throw;
    }
}

/// <summary>
/// Get all the subnets for a Vpc in a set of availability zones.
/// </summary>
/// <param name="vpcId">The Id of the Vpc.</param>
/// <param name="availabilityZones">The list of availability zones.</param>
/// <returns>The collection of subnet objects.</returns>
public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
{
    try
    {
        var subnets = new List<Subnet>();
        var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
            new DescribeSubnetsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new("vpc-id", new List<string>() { vpcId }),
                    new("availability-zone", availabilityZones),
                    new("default-for-az", new List<string>() { "true" })
                }
            });

        // Get the entire list using the paginator.
        await foreach (var subnet in subnetPaginator.Subnets)
        {
            subnets.Add(subnet);
        }
    }
}
```

```
        return subnets;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
        {
            _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"An error occurred while describing the
subnets.: {ex.Message}");
        throw;
    }
}

/// <summary>
/// Delete a launch template by name.
/// </summary>
/// <param name="templateName">The name of the template to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteTemplateByName(string templateName)
{
    try
    {
        await _amazonEc2.DeleteLaunchTemplateAsync(
            new DeleteLaunchTemplateRequest()
            {
                LaunchTemplateName = templateName
            });
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
        {
            _logger.LogError(
                $"Could not delete the template, the name
{_launchTemplateName} was not found.");
        }
    }
}
```

```
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
        throw;
    }
}

/// <summary>
/// Detaches a role from an instance profile, detaches policies from the
role,
/// and deletes all the resources.
/// </summary>
/// <param name="profileName">The name of the profile to delete.</param>
/// <param name="roleName">The name of the role to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteInstanceProfile(string profileName, string roleName)
{
    try
    {
        await _amazonIam.RemoveRoleFromInstanceProfileAsync(
            new RemoveRoleFromInstanceProfileRequest()
            {
                InstanceProfileName = profileName,
                RoleName = roleName
            });
        await _amazonIam.DeleteInstanceProfileAsync(
            new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
        var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
            new ListAttachedRolePoliciesRequest() { RoleName = roleName });
        foreach (var policy in attachedPolicies.AttachedPolicies)
        {
            await _amazonIam.DetachRolePolicyAsync(
                new DetachRolePolicyRequest()
                {
                    RoleName = roleName,
                    PolicyArn = policy.PolicyArn
                });
        }
    }
}
```

```
        // Delete the custom policies only.
        if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
        {
            await _amazonIam.DeletePolicyAsync(
                new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                {
                    PolicyArn = policy.PolicyArn
                });
        }
    }

    await _amazonIam.DeleteRoleAsync(
        new DeleteRoleRequest() { RoleName = roleName });
}
catch (NoSuchEntityException)
{
    Console.WriteLine($"Instance profile {profileName} does not exist.");
}
}

/// <summary>
/// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
/// </summary>
/// <param name="group">The name of the auto scaling group.</param>
/// <returns>A collection of instance Ids.</returns>
public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
{
    var instanceResponse = await
    _amazonAutoScaling.DescribeAutoScalingGroupsAsync(
        new DescribeAutoScalingGroupsRequest()
        {
            AutoScalingGroupNames = new List<string>() { group }
        });
    var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
        g => g.Instances.Select(i => i.InstanceId));
    return instanceIds;
}

/// <summary>
/// Get the instance profile association data for an instance.
/// </summary>
/// <param name="instanceId">The Id of the instance.</param>
/// <returns>Instance profile associations data.</returns>
```



```
public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
{
    try
    {
        var response = await
        _amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
            new DescribeIamInstanceProfileAssociationsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new("instance-id", new List<string>() { instanceId })
                },
            });
        return response.IamInstanceProfileAssociations[0];
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
        {
            _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
        throw;
    }
}

/// <summary>
/// Replace the profile associated with a running instance. After the profile
is replaced, the instance
/// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
/// used to restart the Python web server.
/// </summary>
/// <param name="instanceId">The Id of the instance to update.</param>
/// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
```

```
    /// <param name="associationId">The Id of the existing profile association
    for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
    credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
        }
    }
}
```

```

        Console.WriteLine($"Sending restart command to instance
{instanceId}");
        await _amazonSsm.SendCommandAsync(
            new SendCommandRequest()
            {
                InstanceIds = new List<string>() { instanceId },
                DocumentName = "AWS-RunShellScript",
                Parameters = new Dictionary<string, List<string>>()
                {
                    {
                        "commands",
                        new List<string>() { "cd / && sudo python3 server.py
80" }
                    }
                }
            });
        Console.WriteLine($"Restarted the web server on instance
{instanceId}");
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
        {
            _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
        }

        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
        throw;
    }
}

/// <summary>
/// Try to terminate an instance by its Id.
/// </summary>
/// <param name="instanceId">The Id of the instance to terminate.</param>
/// <returns>Async task.</returns>
public async Task TryTerminateInstanceById(string instanceId)
{

```

```
    var stopping = false;
    Console.WriteLine($"Stopping {instanceId}...");
    while (!stopping)
    {
        try
        {
            await
            _amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                new TerminateInstanceInAutoScalingGroupRequest()
                {
                    InstanceId = instanceId,
                    ShouldDecrementDesiredCapacity = false
                });
            stopping = true;
        }
        catch (ScalingActivityInProgressException)
        {
            Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
            Thread.Sleep(10000);
        }
    }
}

/// <summary>
/// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
/// waits and retries until the group is successfully deleted.
/// </summary>
/// <param name="groupName">The name of the group to try to delete.</param>
/// <returns>Async task.</returns>
public async Task TryDeleteGroupByName(string groupName)
{
    var stopped = false;
    while (!stopped)
    {
        try
        {
            await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                new DeleteAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName
                });
            stopped = true;
        }
    }
}
```

```
    }
    catch (Exception e)
        when ((e is ScalingActivityInProgressException)
            || (e is Amazon.AutoScaling.Model.ResourceInUseException))
    {
        Console.WriteLine($"Some instances are still running.
Waiting...");
        Thread.Sleep(10000);
    }
}

/// <summary>
/// Terminate instances and delete the Auto Scaling group by name.
/// </summary>
/// <param name="groupName">The name of the group to delete.</param>
/// <returns>Async task.</returns>
public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
{
    var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
    new DescribeAutoScalingGroupsRequest()
    {
        AutoScalingGroupNames = new List<string>() { groupName }
    });
    if (describeGroupsResponse.AutoScalingGroups.Any())
    {
        // Update the size to 0.
        await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
            new UpdateAutoScalingGroupRequest()
            {
                AutoScalingGroupName = groupName,
                MinSize = 0
            });
        var group = describeGroupsResponse.AutoScalingGroups[0];
        foreach (var instance in group.Instances)
        {
            await TryTerminateInstanceById(instance.InstanceId);
        }

        await TryDeleteGroupByName(groupName);
    }
    else
```

```
    {
        Console.WriteLine($"No groups found with name {groupName}.");
    }
}

/// <summary>
/// Get the default security group for a specified Vpc.
/// </summary>
/// <param name="vpc">The Vpc to search.</param>
/// <returns>The default security group.</returns>
public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
{
    var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
        new DescribeSecurityGroupsRequest()
        {
            Filters = new List<Amazon.EC2.Model.Filter>()
            {
                new ("group-name", new List<string>() { "default" }),
                new ("vpc-id", new List<string>() { vpc.VpcId })
            }
        });
    return groupResponse.SecurityGroups[0];
}

/// <summary>
/// Verify the default security group of a Vpc allows ingress from the
calling computer.
/// This can be done by allowing ingress from this computer's IP address.
/// In some situations, such as connecting from a corporate network, you must
instead specify
/// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
/// If you do, be sure to remove public access when you're done.
/// </summary>
/// <param name="vpc">The group to check.</param>
/// <param name="port">The port to verify.</param>
/// <param name="ipAddress">This computer's IP address.</param>
/// <returns>True if the ip address is allowed on the group.</returns>
public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
{
    var portIsOpen = false;
    foreach (var ipPermission in group.IpPermissions)
```

```
{
    if (ipPermission.FromPort == port)
    {
        foreach (var ipRange in ipPermission.Ipv4Ranges)
        {
            var cidr = ipRange.CidrIp;
            if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
            {
                portIsOpen = true;
            }
        }

        if (ipPermission.PrefixListIds.Any())
        {
            portIsOpen = true;
        }

        if (!portIsOpen)
        {
            Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                             "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
        }
        else
        {
            break;
        }
    }
}

return portIsOpen;
}

/// <summary>
/// Add an ingress rule to the specified security group that allows access on
the
/// specified port from the specified IP address.
/// </summary>
/// <param name="groupId">The Id of the security group to modify.</param>
/// <param name="port">The port to open.</param>
/// <param name="ipAddress">The IP address to allow access.</param>
/// <returns>Async task.</returns>
public async Task OpenInboundPort(string groupId, int port, string ipAddress)
```

```

    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>

```



```

    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.WriteLine(". ");
        } while (!hasState);

        return hasState;
    }
}

```

Erstellen Sie eine Klasse, die Elastic-Load-Balancing-Aktionen beinhaltet.

```

    /// <summary>
    /// Encapsulates Elastic Load Balancer actions.
    /// </summary>
    public class ElasticLoadBalancerWrapper
    {
        private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
        private string? _endpoint = null;
        private readonly string _targetGroupName = "";
        private readonly string _loadBalancerName = "";
        HttpClient _httpClient = new();
    }

```

```
public string TargetGroupName => _targetGroupName;
public string LoadBalancerName => _loadBalancerName;

/// <summary>
/// Constructor for the Elastic Load Balancer wrapper.
/// </summary>
/// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
client.</param>
/// <param name="configuration">The injected configuration.</param>
public ElasticLoadBalancerWrapper(
    IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
    IConfiguration configuration)
{
    _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
    var prefix = configuration["resourcePrefix"];
    _targetGroupName = prefix + "-tg";
    _loadBalancerName = prefix + "-lb";
}

/// <summary>
/// Get the HTTP Endpoint of a load balancer by its name.
/// </summary>
/// <param name="loadBalancerName">The name of the load balancer.</param>
/// <returns>The HTTP endpoint.</returns>
public async Task<string> GetEndpointForLoadBalancerByName(string
loadBalancerName)
{
    if (_endpoint == null)
    {
        var endpointResponse =
            await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                new DescribeLoadBalancersRequest()
                {
                    Names = new List<string>() { loadBalancerName }
                });
        _endpoint = endpointResponse.LoadBalancers[0].DNSName;
    }

    return _endpoint;
}

/// <summary>
/// Return the GET response for an endpoint as text.
```

```
/// </summary>
/// <param name="endpoint">The endpoint for the request.</param>
/// <returns>The request response.</returns>
public async Task<string> GetEndPointResponse(string endpoint)
{
    var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
    var textResponse = await endpointResponse.Content.ReadAsStringAsync();
    return textResponse!;
}

/// <summary>
/// Get the target health for a group by name.
/// </summary>
/// <param name="groupName">The name of the group.</param>
/// <returns>The collection of health descriptions.</returns>
public async Task<List<TargetHealthDescription>>
CheckTargetHealthForGroup(string groupName)
{
    List<TargetHealthDescription> result = null!;
    try
    {
        var groupResponse =
            await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                new DescribeTargetGroupsRequest()
                {
                    Names = new List<string>() { groupName }
                });
        var healthResponse =
            await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                new DescribeTargetHealthRequest()
                {
                    TargetGroupArn =
groupResponse.TargetGroups[0].TargetGroupArn
                });
        ;
        result = healthResponse.TargetHealthDescriptions;
    }
    catch (TargetGroupNotFoundException)
    {
        Console.WriteLine($"Target group {groupName} not found.");
    }
    return result;
}
```

```
    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
    how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened
    times and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
    unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</
param>
    /// <param name="vpcId">The Id of the Vpc in which the load balancer
    exists.</param>
    /// <returns>The new TargetGroup object.</returns>
    public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
    ProtocolEnum protocol, int port, string vpcId)
    {
        var createResponse = await
    _amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
        new CreateTargetGroupRequest()
        {
            Name = groupName,
            Protocol = protocol,
            Port = port,
            HealthCheckPath = "/healthcheck",
            HealthCheckIntervalSeconds = 10,
            HealthCheckTimeoutSeconds = 5,
            HealthyThresholdCount = 2,
            UnhealthyThresholdCount = 2,
            VpcId = vpcId
        });
        var targetGroup = createResponse.TargetGroups[0];
        return targetGroup;
    }

    /// <summary>
    /// Create an Elastic Load Balancing load balancer that uses the specified
    subnets
    /// and forwards requests to the specified target group.
    /// </summary>
    /// <param name="name">The name for the new load balancer.</param>
```

```
/// <param name="subnetIds">Subnets for the load balancer.</param>
/// <param name="targetGroup">Target group for forwarded requests.</param>
/// <returns>The new LoadBalancer object.</returns>
public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
{
    var createLbResponse = await
    _amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
        new CreateLoadBalancerRequest()
        {
            Name = name,
            Subnets = subnetIds
        });
    var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;

    // Wait for load balancer to be available.
    var loadBalancerReady = false;
    while (!loadBalancerReady)
    {
        try
        {
            var describeResponse =
                await
                _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });

            var loadBalancerState =
                describeResponse.LoadBalancers[0].State.Code;

            loadBalancerReady = loadBalancerState ==
                LoadBalancerStateEnum.Active;
        }
        catch (LoadBalancerNotFoundException)
        {
            loadBalancerReady = false;
        }
        Thread.Sleep(10000);
    }
    // Create the listener.
    await _amazonElasticLoadBalancingV2.CreateListenerAsync(
        new CreateListenerRequest()
```

```
        {
            LoadBalancerArn = loadBalancerArn,
            Protocol = targetGroup.Protocol,
            Port = targetGroup.Port,
            DefaultActions = new List<Action>()
            {
                new Action()
                {
                    Type = ActionTypeEnum.Forward,
                    TargetGroupArn = targetGroup.TargetGroupArn
                }
            }
        });
    return createLbResponse.LoadBalancers[0];
}

/// <summary>
/// Verify this computer can successfully send a GET request to the
/// load balancer endpoint.
/// </summary>
/// <param name="endpoint">The endpoint to check.</param>
/// <returns>True if successful.</returns>
public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
{
    var success = false;
    var retries = 3;
    while (!success && retries > 0)
    {
        try
        {
            var endpointResponse = await _httpClient.GetAsync($"http://{
{endpoint}}");
            Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

            if (endpointResponse.IsSuccessStatusCode)
            {
                success = true;
            }
            else
            {
                retries = 0;
            }
        }
        catch (HttpRequestException)
```

```
        {
            Console.WriteLine("Connection error, retrying...");
            retries--;
            Thread.Sleep(10000);
        }
    }

    return success;
}

/// <summary>
/// Delete a load balancer by its specified name.
/// </summary>
/// <param name="name">The name of the load balancer to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteLoadBalancerByName(string name)
{
    try
    {
        var describeLoadBalancerResponse =
            await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                new DescribeLoadBalancersRequest()
                {
                    Names = new List<string>() { name }
                });
        var lbArn =
describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
        await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
            new DeleteLoadBalancerRequest()
            {
                LoadBalancerArn = lbArn
            }
        );
    }
    catch (LoadBalancerNotFoundException)
    {
        Console.WriteLine($"Load balancer {name} not found.");
    }
}

/// <summary>
/// Delete a TargetGroup by its specified name.
/// </summary>
/// <param name="groupName">Name of the group to delete.</param>
```

```
/// <returns>Async task.</returns>
public async Task DeleteTargetGroupByName(string groupName)
{
    var done = false;
    while (!done)
    {
        try
        {
            var groupResponse =
                await
                _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });

            var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
            await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                new DeleteTargetGroupRequest() { TargetGroupArn =
targetArn });
            Console.WriteLine($"Deleted load balancing target group
{groupName}.");
            done = true;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine(
                $"Target group {groupName} not found, could not delete.");
            done = true;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine("Target group not yet released, waiting...");
            Thread.Sleep(10000);
        }
    }
}
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.


```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books,
/// movies, and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.WriteLine($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                {
                    new AttributeDefinition()
                    {
                        AttributeName = "MediaType",
```

```
        AttributeType = ScalarAttributeType.S
    },
    new AttributeDefinition()
    {
        AttributeName = "ItemId",
        AttributeType = ScalarAttributeType.N
    }
},
KeySpec = new List<KeySchemaElement>()
{
    new KeySchemaElement()
    {
        AttributeName = "MediaTypeId",
        KeyType = KeyType.HASH
    },
    new KeySchemaElement()
    {
        AttributeName = "ItemId",
        KeyType = KeyType.RANGE
    }
},
ProvisionedThroughput = new ProvisionedThroughput()
{
    ReadCapacityUnits = 5,
    WriteCapacityUnits = 5
}
};
await _amazonDynamoDb.CreateTableAsync(createRequest);

// Wait until the table is ACTIVE and then report success.
Console.WriteLine("\nWaiting for table to become active...");

var request = new DescribeTableRequest
{
    TableName = tableName
};

TableStatus status;
do
{
    Thread.Sleep(2000);

    var describeTableResponse = await
        _amazonDynamoDb.DescribeTableAsync(request);
```

```
        status = describeTableResponse.Table.TableStatus;

        Console.WriteLine(".");
    }
    while (status != "ACTIVE");

    return status == TableStatus.ACTIVE;
}
catch (ResourceInUseException)
{
    Console.WriteLine($"Table {tableName} already exists.");
    return false;
}
}

/// <summary>
/// Populate the database table with data from a specified path.
/// </summary>
/// <param name="databaseTableName">The name of the table.</param>
/// <param name="recommendationsPath">The path of the recommendations data.</
param>
/// <returns>Async task.</returns>
public async Task PopulateDatabase(string databaseTableName, string
recommendationsPath)
{
    var recommendationsText = await
File.ReadAllTextAsync(recommendationsPath);
    var records =

JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
    var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

    foreach (var record in records!)
    {
        batchWrite.AddPutItem(record);
    }

    await batchWrite.ExecuteAsync();
}

/// <summary>
/// Delete the recommendation table by name.
/// </summary>
/// <param name="tableName">The name of the recommendation table.</param>
```

```

    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}

```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```

    /// <summary>
    /// Encapsulates Systems Manager parameter operations. This example uses these
    /// parameters
    /// to drive the demonstration of resilient architecture, such as failure of a
    /// dependency or
    /// how the service responds to a health check.
    /// </summary>
    public class SmParameterWrapper
    {
        private readonly IAmazonSimpleSystemsManagement
            _amazonSimpleSystemsManagement;

        private readonly string _tableParameter = "doc-example-resilient-
            architecture-table";
        private readonly string _failureResponseParameter = "doc-example-resilient-
            architecture-failure-response";
        private readonly string _healthCheckParameter = "doc-example-resilient-
            architecture-health-check";
        private readonly string _tableName = "";

        public string TableParameter => _tableParameter;
        public string TableName => _tableName;
        public string HealthCheckParameter => _healthCheckParameter;
        public string FailureResponseParameter => _failureResponseParameter;
    }

```

```
/// <summary>
/// Constructor for the SmParameterWrapper.
/// </summary>
/// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
Management client.</param>
/// <param name="configuration">The injected configuration.</param>
public SmParameterWrapper(IAmazonSimpleSystemsManagement
amazonSimpleSystemsManagement, IConfiguration configuration)
{
    _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
    _tableName = configuration["databaseName"]!;
}

/// <summary>
/// Reset the Systems Manager parameters to starting values for the demo.
/// </summary>
/// <returns>Async task.</returns>
public async Task Reset()
{
    await this.PutParameterByName(_tableParameter, _tableName);
    await this.PutParameterByName(_failureResponseParameter, "none");
    await this.PutParameterByName(_healthCheckParameter, "shallow");
}


/// <summary>
/// Set the value of a named Systems Manager parameter.
/// </summary>
/// <param name="name">The name of the parameter.</param>
/// <param name="value">The value to set.</param>
/// <returns>Async task.</returns>
public async Task PutParameterByName(string name, string value)
{
    await _amazonSimpleSystemsManagement.PutParameterAsync(
        new PutParameterRequest() { Name = name, Value = value, Overwrite =
true });
}
}
```

- APIEinzelheiten finden Sie in den folgenden Themen in der AWS SDK for .NET APIReferenz.
 - [AttachLoadBalancerTargetGroups](#)

- [CreateAutoScalingGroup](#)
- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacesIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu. [GitHub](#) Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
public class Main {

    public static final String fileName = "C:\\\\AWS\\\\resworkflow\\
\\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\\\AWS\\\\resworkflow\\
\\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\\\AWS\\\\resworkflow\\
\\instance_policy.json"; // Modify file location.
    public static final String ssmJSON = "C:\\\\AWS\\\\resworkflow\\
\\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-
prof-bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;
```

```
public static final String DASHES = new String(new char[80]).replace("\0",
"-");

public static void main(String[] args) throws IOException,
InterruptedException {
    Scanner in = new Scanner(System.in);
    Database database = new Database();
    AutoScaler autoScaler = new AutoScaler();
    LoadBalancer loadBalancer = new LoadBalancer();

    System.out.println(DASHES);
    System.out.println("Welcome to the demonstration of How to Build and
Manage a Resilient Service!");
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("A - SETUP THE RESOURCES");
    System.out.println("Press Enter when you're ready to start deploying
resources.");
    in.nextLine();
    deploy(loadBalancer);
    System.out.println(DASHES);
    System.out.println(DASHES);
    System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    demo(loadBalancer);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("C - DELETE THE RESOURCES");
    System.out.println("""
        This concludes the demo of how to build and manage a resilient
service.

        To keep things tidy and to avoid unwanted charges on your
account, we can clean up all AWS resources
that were created for this demo.
        """);

    System.out.println("\n Do you want to delete the resources (y/n)? ");
    String userInput = in.nextLine().trim().toLowerCase(); // Capture user
input

    if (userInput.equals("y")) {
```



```
        // Delete resources here
        deleteResources(loadBalancer, autoScaler, database);
        System.out.println("Resources deleted.");
    } else {
        System.out.println("""
            Okay, we'll leave the resources intact.
            Don't forget to delete them when you're done with them or you
might incur unexpected charges.
            """);
    }
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("The example has completed. ");
    System.out.println("\n Thanks for watching!");
    System.out.println(DASHES);
}

// Deletes the AWS resources used in this example.
private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
    throws IOException, InterruptedException {
    loadBalancer.deleteLoadBalancer(lbName);
    System.out.println("*** Wait 30 secs for resource to be deleted");
    TimeUnit.SECONDS.sleep(30);
    loadBalancer.deleteTargetGroup(targetGroupName);
    autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
    autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
    autoScaler.deleteTemplate(templateName);
    database.deleteTable(tableName);
}

private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
    Scanner in = new Scanner(System.in);
    System.out.println(
        """

            For this demo, we'll use the AWS SDK for Java (v2) to
create several AWS resources
            to set up a load-balanced web service endpoint and
explore some ways to make it resilient
            against various kinds of failures.

            Some of the resources create by this demo are:
```

```
        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
        \t* An EC2 launch template that defines EC2 instances
that each contain a Python web server.
        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
        """);

    System.out.println("Press Enter when you're ready.");
    in.nextLine();
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("Creating and populating a DynamoDB table named " +
tableName);
    Database database = new Database();
    database.createTable(tableName, fileName);
    System.out.println(DASHES);

    System.out.println(DASHES);
    System.out.println("""
        Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.
        This script starts a Python web server defined in the `server.py`
script. The web server
        listens to HTTP requests on port 80 and responds to requests to
`/` and to `/healthcheck`.
        For demo purposes, this server is run as the root user. In
production, the best practice is to
        run a web server, such as Apache, with least-privileged
credentials.

        The template also defines an IAM policy that each instance uses
to assume a role that grants
        permissions to access the DynamoDB recommendation table and
Systems Manager parameters
        that control the flow of the demo.
        """);

    LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
    templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
```

```
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println(
    "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
System.out.println("*** Wait 30 secs for the VPC to be created");
TimeUnit.SECONDS.sleep(30);
AutoScaler autoScaler = new AutoScaler();
String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

System.out.println("""
    At this point, you have EC2 instances created. Once each instance
starts, it listens for
    HTTP requests. You can see these instances in the console or
continue with the demo.
    Press Enter when you're ready to continue.
    """);

in.nextLine();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("Creating variables that control the flow of the
demo.");
ParameterHelper paramHelper = new ParameterHelper();
paramHelper.reset();
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("""
    Creating an Elastic Load Balancing target group and load
balancer. The target group
    defines how the load balancer connects to instances. The load
balancer provides a
    single endpoint where clients connect and dispatches requests to
instances in the group.
    """);

String vpcId = autoScaler.getDefaultVPC();
List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
```

```
String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
String elbDnsName = loadBalancer.createLoadBalancer(subnets,
targetGroupArn, lbName, port, protocol);
autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
System.out.println("Verifying access to the load balancer endpoint...");
boolean wasSuccessful =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
if (!wasSuccessful) {
    System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
    CloseableHttpClient httpClient = HttpClients.createDefault();

    // Create an HTTP GET request to "http://checkip.amazonaws.com"
    HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
    try {
        // Execute the request and get the response
        HttpResponse response = httpClient.execute(httpGet);

        // Read the response content.
        String ipAddress =
IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8).trim();

        // Print the public IP address.
        System.out.println("Public IP Address: " + ipAddress);
        GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
        if (!groupInfo.isPortOpen()) {
            System.out.println("""
                For this example to work, the default security group
for your default VPC must
                allow access from this computer. You can either add
it automatically from this
                example or add it yourself using the AWS Management
Console.
                """);

            System.out.println(
                "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
            System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
```

```
        String ans = in.nextLine();
        if ("y".equalsIgnoreCase(ans)) {
            autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
            System.out.println("Security group rule added.");
        } else {
            System.out.println("No security group rule added.");
        }
    }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
} else if (wasSuccessul) {
    System.out.println("Your load balancer is ready. You can access it by
browsing to:");
    System.out.println("\t http://" + elbDnsName);
} else {
    System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
    System.out.println("manually verifying that your VPC and security
group are configured correctly and that");
    System.out.println("you can successfully make a GET request to the
load balancer.");
}

    System.out.println("Press Enter when you're ready to continue with the
demo.");
    in.nextLine();
}

// A method that controls the demo part of the Java program.
public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
    ParameterHelper paramHelper = new ParameterHelper();
    System.out.println("Read the ssm_only_policy.json file");
    String ssmOnlyPolicy = readFileAsString(ssmJSON);

    System.out.println("Resetting parameters to starting values for demo.");
    paramHelper.reset();

    System.out.println(
        """"
```

This part of the demonstration shows how to toggle different parts of the system to create situations where the web service fails, and shows how using a resilient architecture can keep the web service running in spite of these failures.

At the start, the load balancer endpoint returns recommendations and reports that all targets are healthy.

```
        """);  
demoChoices(loadBalancer);
```

```
System.out.println(  
    ""
```

The web service running on the EC2 instances gets recommendations by querying a DynamoDB table.

The table name is contained in a Systems Manager parameter named `self.param_helper.table`.

To simulate a failure of the recommendation service, let's set this parameter to name a non-existent table.

```
        """);  
paramHelper.put(paramHelper.tableName, "this-is-not-a-table");
```

```
System.out.println(  
    ""
```

\nNow, sending a GET request to the load balancer endpoint returns a failure code. But, the service reports as healthy to the load balancer because shallow health checks don't check for failure of the recommendation service.

```
        """);  
demoChoices(loadBalancer);
```

```
System.out.println(  
    ""
```

Instead of failing when the recommendation service fails, the web service can return a static response.

While this is not a perfect solution, it presents the customer with a somewhat better experience than failure.

```
        """);  
paramHelper.put(paramHelper.failureResponse, "static");
```

```
System.out.println("""
```

Now, sending a GET request to the load balancer endpoint returns a static response.

```
        The service still reports as healthy because health checks are
still shallow.
        """);
demoChoices(loadBalancer);

System.out.println("Let's reinstate the recommendation service.");
paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

System.out.println("""
        Let's also substitute bad credentials for one of the instances in
the target group so that it can't
        access the DynamoDB recommendation table. We will get an instance
id value.
        """);

LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
AutoScaler autoScaler = new AutoScaler();

// Create a new instance profile based on badCredsProfileName.
templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

String profileAssociationId =
autoScaler.getInstanceProfile(badInstanceId);
System.out.println("The association Id value is " +
profileAssociationId);
System.out.println("Replacing the profile for instance " + badInstanceId
        + " with a profile that contains bad credentials");
autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

System.out.println(
        ""
        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
        depending on which instance is selected by the load
balancer.
        """);

demoChoices(loadBalancer);
```


Even while the instance is terminating and the new instance is starting, sending a GET request to the web service continues to get a successful recommendation response because the load balancer routes requests to the healthy instances. After the replacement instance starts and reports as healthy, it is included in the load balancing rotation.

Note that terminating and replacing an instance typically takes several minutes, during which time you can see the changing health check status until the new instance is running and healthy.

```
        """);

        demoChoices(loadBalancer);
        System.out.println(
            "If the recommendation service fails now, deep health checks mean
            all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
    }

    public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
    InterruptedException {
        String[] actions = {
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo."
        };

        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("-".repeat(88));
            System.out.println("See the current state of the service by selecting
            one of the following choices:");
            for (int i = 0; i < actions.length; i++) {
                System.out.println(i + ": " + actions[i]);
            }

            try {
                System.out.print("\nWhich action would you like to take? ");
                int choice = scanner.nextInt();
```

```
System.out.println("-".repeat(88));

switch (choice) {
    case 0 -> {
        System.out.println("Request:\n");
        System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
        CloseableHttpClient httpClient =
HttpClientClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode =
response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);

        // Display the JSON response
        BufferedReader reader = new BufferedReader(
            new
InputStreamReader(response.getEntity().getContent()));
        StringBuilder jsonResponse = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            jsonResponse.append(line);
        }
        reader.close();

        // Print the formatted JSON response.
        System.out.println("Full Response:\n");
        System.out.println(jsonResponse.toString());

        // Close the HTTP client.
        httpClient.close();
    }
    case 1 -> {
        System.out.println("\nChecking the health of load
balancer targets:\n");
        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
```

```

        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
health check to update
                                Note that it can take a minute or two for the
                                after changes are made.
                                """);
    }
    case 2 -> {
        System.out.println("\nOkay, let's move on.");
        System.out.println("-".repeat(88));
        return; // Exit the method when choice is 2
    }
    default -> System.out.println("You must choose a value
between 0-2. Please select again.");
}

    } catch (java.util.InputMismatchException e) {
        System.out.println("Invalid input. Please select again.");
        scanner.nextLine(); // Clear the input buffer.
    }
}

public static String readFileAsString(String filePath) throws IOException {
    byte[] bytes = Files.readAllBytes(Paths.get(filePath));
    return new String(bytes);
}
}

```

Erstellen Sie eine Klasse, die Auto Scaling- und EC2 Amazon-Aktionen umschließt.

```

public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;
}

```

```
private static SsmClient ssmClient;

private iamClient getIAMClient() {
    if (iamClient == null) {
        iamClient = iamClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return iamClient;
}

private SsmClient getSSMClient() {
    if (ssmClient == null) {
        ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ssmClient;
}

private Ec2Client getEc2Client() {
    if (ec2Client == null) {
        ec2Client = Ec2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return ec2Client;
}

private AutoScalingClient getAutoScalingClient() {
    if (autoScalingClient == null) {
        autoScalingClient = AutoScalingClient.builder()
            .region(Region.US_EAST_1)
            .build();
    }
    return autoScalingClient;
}

/**
 * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
 * terminated, it can no longer be accessed.
 */
public void terminateInstance(String instanceId) {
```

```
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
            .builder()
            .instanceId(instanceId)
            .shouldDecrementDesiredCapacity(false)
            .build();

getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
    System.out.format("Terminated instance %s.", instanceId);
}

/**
 * Replaces the profile associated with a running instance. After the profile
is
 * replaced, the instance is rebooted to ensure that it uses the new profile.
 * When
 * the instance is ready, Systems Manager is used to restart the Python web
 * server.
 */
public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
    throws InterruptedException {
    // Create an IAM instance profile specification.
    software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
    .builder()
    .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
    // name.

    .build();

    // Replace the IAM instance profile association for the EC2 instance.
    ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
    .builder()
    .iamInstanceProfile(iamInstanceProfile)
    .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.

    .build();

    try {
        getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
```

```
        // Handle the response as needed.
    } catch (Ec2Exception e) {
        // Handle exceptions, log, or report the error.
        System.err.println("Error: " + e.getMessage());
    }
    System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
        newInstanceProfileName);
    TimeUnit.SECONDS.sleep(15);
    boolean instReady = false;
    int tries = 0;

    // Reboot after 60 seconds
    while (!instReady) {
        if (tries % 6 == 0) {
            getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                .instanceIds(instanceId)
                .build());
            System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
        }
        tries++;
        try {
            TimeUnit.SECONDS.sleep(10);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
        List<InstanceInformation> instanceInformationList =
informationResponse.getInstanceInformationList();
        for (InstanceInformation info : instanceInformationList) {
            if (info.getInstanceId().equals(instanceId)) {
                instReady = true;
                break;
            }
        }
    }

    SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
        .instanceIds(instanceId)
        .documentName("AWS-RunShellScript")
        .parameters(Collections.singletonMap("commands",
```

```
        Collections.singletonList("cd / && sudo python3 server.py
80"))))
        .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress)
    {
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
            .groupName(secGroupId)
            .cidrIp(ipAddress)
            .fromPort(Integer.parseInt(port))
            .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }

    /**
     * Detaches a role from an instance profile, detaches policies from the role,
     * and deletes all the resources.
     */
    public void deleteInstanceProfile(String roleName, String profileName) {
        try {
            software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
            .builder()
            .instanceProfileName(profileName)
            .build();

            GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
            String name = response.instanceProfile().instanceProfileName();
            System.out.println(name);

            RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                .instanceProfileName(profileName)
```

```
        .roleName(roleName)
        .build();

        getIAMClient().removeRoleFromInstanceProfile(profileRequest);
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
        .instanceProfileName(profileName)
        .build();

        getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
        System.out.println("Deleted instance profile " + profileName);

        DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
        .roleName(roleName)
        .build();

        // List attached role policies.
        ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
        .listAttachedRolePolicies(role -> role.roleName(roleName));
        List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
        .roleName(roleName)
        .policyArn(attachedPolicy.policyArn())
        .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
        }

        getIAMClient().deleteRole(deleteRoleRequest);
        System.out.println("Instance profile and role deleted.");

    } catch (IamException e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

public void deleteTemplate(String templateName) {
```



```
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
            .autoScalingGroupName(groupName)
            .forceDelete(true)
            .build();

getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /**
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network,
you
     * must instead specify a prefix list ID. You can also temporarily open the
port
     * to
     * any IP address while running this example. If you do, be sure to remove
     * public
     * access when you're done.
     *
     */
    public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
        boolean portIsOpen = false;
        GroupInfo groupInfo = new GroupInfo();
        try {
            Filter filter = Filter.builder()
                .name("group-name")
                .values("default")
                .build();

            Filter filter1 = Filter.builder()
                .name("vpc-id")
                .values(VPC)
                .build();
```

```
DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
    .filters(filter, filter1)
    .build();

DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
    .describeSecurityGroups(securityGroupsRequest);
String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
groupInfo.setGroupName(securityGroup);

for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
    System.out.println("Found security group: " +
secGroup.groupId());

    for (IpPermission ipPermission : secGroup.ipPermissions()) {
        if (ipPermission.fromPort() == port) {
            System.out.println("Found inbound rule: " +
ipPermission);

            for (IpRange ipRange : ipPermission.ipRanges()) {
                String cidrIp = ipRange.cidrIp();
                if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                    System.out.println(cidrIp + " is applicable");
                    portIsOpen = true;
                }
            }

            if (!ipPermission.prefixListIds().isEmpty()) {
                System.out.println("Prefix lList is applicable");
                portIsOpen = true;
            }

            if (!portIsOpen) {
                System.out
                    .println("The inbound rule does not appear to
be open to either this computer's IP,"
                    + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
            } else {
                break;
            }
        }
    }
}
```

```
        }
    }
}

} catch (AutoScalingException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}

groupInfo.setPortOpen(portIsOpen);
return groupInfo;
}

/*
 * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 * Scaling group.
 * The target group specifies how the load balancer forward requests to the
 * instances
 * in the group.
 */
public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
    try {
        AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
            .autoScalingGroupName(asGroupName)
            .targetGroupARNs(targetGroupARN)
            .build();

getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
        System.out.println("Attached load balancer to " + asGroupName);

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

// Creates an EC2 Auto Scaling group with the specified size.
public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

    // Get availability zones.
```

```
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
    .builder()
    .build();

DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
    .collect(Collectors.toList());

String availabilityZones = String.join(",", availabilityZoneNames);
LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
    .launchTemplateName(templateName)
    .version("$Default")
    .build();

String[] zones = availabilityZones.split(",");
CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
    .launchTemplate(specification)
    .availabilityZones(zones)
    .maxSize(groupSize)
    .minSize(groupSize)
    .autoScalingGroupName(autoScalingGroupName)
    .build();

try {
    getAutoScalingClient().createAutoScalingGroup(groupRequest);

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

System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
return zones;
}
```

```
public String getDefaultVPC() {
    // Define the filter.
    Filter defaultFilter = Filter.builder()
        .name("is-default")
        .values("true")
        .build();

    software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
        .builder()
        .filters(defaultFilter)
        .build();

    DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
    return response.vpcs().get(0).vpcId();
}

// Gets the default subnets in a VPC for a specified list of Availability
Zones.
public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
    List<Subnet> subnets = null;
    Filter vpcFilter = Filter.builder()
        .name("vpc-id")
        .values(vpcId)
        .build();

    Filter azFilter = Filter.builder()
        .name("availability-zone")
        .values(availabilityZones)
        .build();

    Filter defaultForAZ = Filter.builder()
        .name("default-for-az")
        .values("true")
        .build();

    DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
        .filters(vpcFilter, azFilter, defaultForAZ)
        .build();

    DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
    subnets = response.subnets();
    return subnets;
}
```

```
}

// Gets data about the instances in the EC2 Auto Scaling group.
public String getBadInstance(String groupName) {
    DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
        .autoScalingGroupNames(groupName)
        .build();

    DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
    AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
    List<String> instanceIds = autoScalingGroup.instances().stream()
        .map(instance -> instance.instanceId())
        .collect(Collectors.toList());

    String[] instanceIdArray = instanceIds.toArray(new String[0]);
    for (String instanceId : instanceIdArray) {
        System.out.println("Instance ID: " + instanceId);
        return instanceId;
    }
    return "";
}

// Gets data about the profile associated with an instance.
public String getInstanceProfile(String instanceId) {
    Filter filter = Filter.builder()
        .name("instance-id")
        .values(instanceId)
        .build();

    DescribeIamInstanceProfileAssociationsRequest associationsRequest =
DescribeIamInstanceProfileAssociationsRequest
        .builder()
        .filters(filter)
        .build();

    DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
        .describeIamInstanceProfileAssociations(associationsRequest);
    return response.iamInstanceProfileAssociations().get(0).associationId();
}

public void deleteRolesPolicies(String policyName, String roleName, String
InstanceProfile) {
```

```
ListPoliciesRequest listPoliciesRequest =
ListPoliciesRequest.builder().build();
ListPoliciesResponse listPoliciesResponse =
getIAMClient().listPolicies(listPoliciesRequest);
for (Policy policy : listPoliciesResponse.policies()) {
    if (policy.policyName().equals(policyName)) {
        // List the entities (users, groups, roles) that are attached to
the policy.

software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
listEntitiesRequest =
software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
    .builder()
    .policyArn(policy.arn())
    .build();
ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
    .listEntitiesForPolicy(listEntitiesRequest);
if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
    || !listEntitiesResponse.policyRoles().isEmpty()) {
    // Detach the policy from any entities it is attached to.
DetachRolePolicyRequest detachPolicyRequest =
DetachRolePolicyRequest.builder()
    .policyArn(policy.arn())
    .roleName(roleName) // Specify the name of the IAM
role

    .build();

    getIAMClient().detachRolePolicy(detachPolicyRequest);
    System.out.println("Policy detached from entities.");
}

// Now, you can delete the policy.
DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
    .policyArn(policy.arn())
    .build();

getIAMClient().deletePolicy(deletePolicyRequest);
System.out.println("Policy deleted successfully.");
break;
}
}
```

```

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
        .roleName(roleName)
        .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .roleName(roleName) // Remove the extra dot here
                .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance
profile " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
            .instanceProfileName(InstanceProfile)
            .build();

        getIAMClient().deleteInstanceProfile(r ->
r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}

```

Erstellen Sie eine Klasse, die Elastic-Load-Balancing-Aktionen beinhaltet.

```

public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {

```



```
        elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }

    return elasticLoadBalancingV2Client;
}

// Checks the health of the instances in the target group.
public List<TargetHealthDescription> checkTargetHealth(String
targetGroupName) {
    DescribeTargetGroupsRequest targetGroupsRequest =
DescribeTargetGroupsRequest.builder()
        .names(targetGroupName)
        .build();

    DescribeTargetGroupsResponse tgResponse =
getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

    DescribeTargetHealthRequest healthRequest =
DescribeTargetHealthRequest.builder()

.targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
        .build();

    DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
    return healthResponse.targetHealthDescriptions();
}

// Gets the HTTP endpoint of the load balancer.
public String getEndpoint(String lbName) {
    DescribeLoadBalancersResponse res = getLoadBalancerClient()
        .describeLoadBalancers(describe -> describe.names(lbName));
    return res.loadBalancers().get(0).dnsName();
}

// Deletes a load balancer.
public void deleteLoadBalancer(String lbName) {
    try {
        // Use a waiter to delete the Load Balancer.
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
            .describeLoadBalancers(describe -> describe.names(lbName));
```

```
        ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
        DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
        .build();

        getLoadBalancerClient().deleteLoadBalancer(
            builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancersDeleted(request);
        waiterResponse.matched().response().ifPresent(System.out::println);

    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(lbName + " was deleted.");
}

// Deletes the target group.
public void deleteTargetGroup(String targetGroupName) {
    try {
        DescribeTargetGroupsResponse res = getLoadBalancerClient()
            .describeTargetGroups(describe ->
describe.names(targetGroupName));
        getLoadBalancerClient()
            .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
    } catch (ElasticLoadBalancingV2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println(targetGroupName + " was deleted.");
}

// Verify this computer can successfully send a GET request to the load
balancer
// endpoint.
public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
    boolean success = false;
    int retries = 3;
```

```
CloseableHttpClient httpClient = HttpClients.createDefault();

// Create an HTTP GET request to the ELB.
HttpGet httpGet = new HttpGet("http://" + elbDnsName);
try {
    while ((!success) && (retries > 0)) {
        // Execute the request and get the response.
        HttpResponse response = httpClient.execute(httpGet);
        int statusCode = response.getStatusLine().getStatusCode();
        System.out.println("HTTP Status Code: " + statusCode);
        if (statusCode == 200) {
            success = true;
        } else {
            retries--;
            System.out.println("Got connection error from load balancer
endpoint, retrying...");
            TimeUnit.SECONDS.sleep(15);
        }
    }

    } catch (org.apache.http.conn.HttpHostConnectException e) {
        System.out.println(e.getMessage());
    }

    System.out.println("Status.." + success);
    return success;
}

/**
 * Creates an Elastic Load Balancing target group. The target group specifies
 * how
 * the load balancer forward requests to instances in the group and how
instance
 * health is checked.
 */
public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
    CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
        .healthCheckPath("/healthcheck")
        .healthCheckTimeoutSeconds(5)
        .port(port)
        .vpcId(vpcId)
        .name(targetGroupName)
```

```
        .protocol(protocol)
        .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /**
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
        String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
                .map(Subnet::subnetId)
                .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                .subnets(subnetIdStrings)
                .name(lbName)
                .scheme("internet-facing")
                .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                .loadBalancerArns(lbARN)
```

```
        .build();

        System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
        WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
            .waitUntilLoadBalancerAvailable(request);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Load Balancer " + lbName + " is available.");

        // Get the DNS name (endpoint) of the load balancer.
        String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
        System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

        // Create a listener for the load balance.
        Action action = Action.builder()
            .targetGroupArn(targetGroupARN)
            .type("forward")
            .build();

        CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

            .loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
                .defaultActions(action)
                .port(port)
                .protocol(protocol)
                .defaultActions(action)
                .build();

        getLoadBalancerClient().createListener(listenerRequest);
        System.out.println("Created listener to forward traffic from load
balancer " + lbName + " to target group "
            + targetGroupARN);

        // Return the load balancer DNS name.
        return lbDNSName;

    } catch (ElasticLoadBalancingV2Exception e) {
        e.printStackTrace();
    }
    return "";
}
}
```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                .region(Region.US_EAST_1)
                .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " +
e.getMessage());
        }
        return false;
    }

    /*
    * Creates a DynamoDB table to use a recommendation service. The table has a
```

```
* hash key named 'MediaType' that defines the type of media recommended,
such
* as
* Book or Movie, and a range key named 'ItemId' that, combined with the
* MediaType,
* forms a unique identifier for the recommended item.
*/
public void createTable(String tableName, String fileName) throws IOException
{
    // First check to see if the table exists.
    boolean doesExist = doesTableExist(tableName);
    if (!doesExist) {
        DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
        CreateTableRequest createTableRequest = CreateTableRequest.builder()
            .tableName(tableName)
            .attributeDefinitions(
                AttributeDefinition.builder()
                    .attributeName("MediaType")
                    .attributeType(ScalarAttributeType.S)
                    .build(),
                AttributeDefinition.builder()
                    .attributeName("ItemId")
                    .attributeType(ScalarAttributeType.N)
                    .build())
            .keySchema(
                KeySchemaElement.builder()
                    .attributeName("MediaType")
                    .keyType(KeyType.HASH)
                    .build(),
                KeySchemaElement.builder()
                    .attributeName("ItemId")
                    .keyType(KeyType.RANGE)
                    .build())
            .provisionedThroughput(
                ProvisionedThroughput.builder()
                    .readCapacityUnits(5L)
                    .writeCapacityUnits(5L)
                    .build())
            .build();

        getDynamoDbClient().createTable(createTableRequest);
        System.out.println("Creating table " + tableName + "...");

        // Wait until the Amazon DynamoDB table is created.
    }
}
```

```
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
            .tableName(tableName)
            .build();

        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Table " + tableName + " created.");

        // Add records to the table.
        populateTable(fileName, tableName);
    }
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws
IOException {
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
        .dynamoDbClient(getDynamoDbClient())
        .build();
    ObjectMapper objectMapper = new ObjectMapper();
    File jsonFile = new File(fileName);
    JsonNode rootNode = objectMapper.readTree(jsonFile);

    DynamoDbTable<Recommendation> mappedTable =
enhancedClient.table(tableName,
        TableSchema.fromBean(Recommendation.class));
    for (JsonNode currentNode : rootNode) {
        String mediaType = currentNode.path("MediaType").path("S").asText();
        int itemId = currentNode.path("ItemId").path("N").asInt();
        String title = currentNode.path("Title").path("S").asText();
        String creator = currentNode.path("Creator").path("S").asText();

        // Create a Recommendation object and set its properties.
        Recommendation rec = new Recommendation();
        rec.setMediaType(mediaType);
        rec.setItemId(itemId);
        rec.setTitle(title);
    }
}
```



```
        rec.setCreator(creator);

        // Put the item into the DynamoDB table.
        mappedTable.putItem(rec); // Add the Recommendation to the list.
    }
    System.out.println("Added all records to the " + tableName);
}
}
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-
response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }

    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
            .region(Region.US_EAST_1)
            .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
            .name(name)
            .value(value)
            .overwrite(true)
            .type("String")
            .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- APIEinzelheiten finden Sie in den folgenden Themen in der AWS SDK for Java 2.x APIReferenz.
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)
 - [CreateInstanceProfile](#)
 - [CreateLaunchTemplate](#)
 - [CreateListener](#)
 - [CreateLoadBalancer](#)
 - [CreateTargetGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DeleteInstanceProfile](#)
 - [DeleteLaunchTemplate](#)
 - [DeleteLoadBalancer](#)
 - [DeleteTargetGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAvailabilityZones](#)
 - [DescribeIamInstanceProfileAssociations](#)
 - [DescribeInstances](#)
 - [DescribeLoadBalancers](#)
 - [DescribeSubnets](#)
 - [DescribeTargetGroups](#)
 - [DescribeTargetHealth](#)
 - [DescribeVpcs](#)
 - [RebootInstances](#)
 - [ReplacelamInstanceProfileAssociation](#)
 - [TerminateInstanceInAutoScalingGroup](#)
 - [UpdateAutoScalingGroup](#)

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 * - deploy
 * - demo
 * - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
```

```

*/
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes] [-v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}

```

Erstellen Sie Schritte, um alle Ressourcen bereitzustellen.

```

import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,

```

```
    DescribeSecurityGroupsCommand,  
    AuthorizeSecurityGroupIngressCommand,  
  } from "@aws-sdk/client-ec2";  
import {  
  IAMClient,  
  CreatePolicyCommand,  
  CreateRoleCommand,  
  CreateInstanceProfileCommand,  
  AddRoleToInstanceProfileCommand,  
  AttachRolePolicyCommand,  
  waitUntilInstanceProfileExists,  
} from "@aws-sdk/client-iam";  
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";  
import {  
  CreateAutoScalingGroupCommand,  
  AutoScalingClient,  
  AttachLoadBalancerTargetGroupsCommand,  
} from "@aws-sdk/client-auto-scaling";  
import {  
  CreateListenerCommand,  
  CreateLoadBalancerCommand,  
  CreateTargetGroupCommand,  
  ElasticLoadBalancingV2Client,  
  waitUntilLoadBalancerAvailable,  
} from "@aws-sdk/client-elastic-load-balancing-v2";  
  
import {  
  ScenarioOutput,  
  ScenarioInput,  
  ScenarioAction,  
} from "@aws-doc-sdk-examples/lib/scenario/index.js";  
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";  
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";  
  
import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";  
import { initParamsSteps } from "./steps-reset-params.js";  
  
/**  
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[][]}  
 */  
export const deploySteps = [  
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),  
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {  
    type: "confirm",
```

```
    }),
    new ScenarioAction(
      "handleConfirmDeployment",
      (c) => c.confirmDeployment === false && process.exit(),
    ),
    new ScenarioOutput(
      "creatingTable",
      MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
    ),
    new ScenarioAction("createTable", async () => {
      const client = new DynamoDBClient({});
      await client.send(
        new CreateTableCommand({
          TableName: NAMES.tableName,
          ProvisionedThroughput: {
            ReadCapacityUnits: 5,
            WriteCapacityUnits: 5,
          },
          AttributeDefinitions: [
            {
              AttributeName: "MediaType",
              AttributeType: "S",
            },
            {
              AttributeName: "ItemId",
              AttributeType: "N",
            },
          ],
          KeySchema: [
            {
              AttributeName: "MediaType",
              KeyType: "HASH",
            },
            {
              AttributeName: "ItemId",
              KeyType: "RANGE",
            },
          ],
        })
      );
      await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
    }),
    new ScenarioOutput(
      "createdTable",
```

```

    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),

```

```
new ScenarioOutput(
  "createdKeyPair",
  MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
),
new ScenarioOutput(
  "creatingInstancePolicy",
  MESSAGES.creatingInstancePolicy.replace(
    "${INSTANCE_POLICY_NAME}",
    NAMES.instancePolicyName,
  ),
),
new ScenarioAction("createInstancePolicy", async (state) => {
  const client = new IAMClient({});
  const {
    Policy: { Arn },
  } = await client.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.instancePolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "instance_policy.json"),
      ),
    }),
  ),
  state.instancePolicyArn = Arn;
}),
new ScenarioOutput("createdInstancePolicy", (state) =>
  MESSAGES.createdInstancePolicy
    .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
    .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
),
new ScenarioOutput(
  "creatingInstanceRole",
  MESSAGES.creatingInstanceRole.replace(
    "${INSTANCE_ROLE_NAME}",
    NAMES.instanceRoleName,
  ),
),
new ScenarioAction("createInstanceRole", () => {
  const client = new IAMClient({});
  return client.send(
    new CreateRoleCommand({
      RoleName: NAMES.instanceRoleName,
      AssumeRolePolicyDocument: readFileSync(
        join(ROOT, "assume-role-policy.json"),
      ),
    }),
  ),
});
```



```
    ),
  }),
);
}),
new ScenarioOutput(
  "createdInstanceRole",
  MESSAGES.createdInstanceRole.replace(
    "${INSTANCE_ROLE_NAME}",
    NAMES.instanceRoleName,
  ),
),
),
new ScenarioOutput(
  "attachingPolicyToRole",
  MESSAGES.attachingPolicyToRole
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
    .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
),
new ScenarioAction("attachPolicyToRole", async (state) => {
  const client = new IAMClient({});
  await client.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.instanceRoleName,
      PolicyArn: state.instancePolicyArn,
    }),
  );
}),
new ScenarioOutput(
  "attachedPolicyToRole",
  MESSAGES.attachedPolicyToRole
    .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
),
new ScenarioOutput(
  "creatingInstanceProfile",
  MESSAGES.creatingInstanceProfile.replace(
    "${INSTANCE_PROFILE_NAME}",
    NAMES.instanceProfileName,
  ),
),
),
new ScenarioAction("createInstanceProfile", async (state) => {
  const client = new IAMClient({});
  const {
    InstanceProfile: { Arn },
  } = await client.send(
```

```

    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.instanceProfileName,
    }),
  );
  state.instanceProfileArn = Arn;

  await waitUntilInstanceProfileExists(
    { client },
    { InstanceProfileName: NAMES.instanceProfileName },
  );
}),
new ScenarioOutput("createdInstanceProfile", (state) =>
  MESSAGES.createdInstanceProfile
    .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
    .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
),
new ScenarioOutput(
  "addingRoleToInstanceProfile",
  MESSAGES.addingRoleToInstanceProfile
    .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
),
new ScenarioAction("addRoleToInstanceProfile", () => {
  const client = new IAMClient({});
  return client.send(
    new AddRoleToInstanceProfileCommand({
      RoleName: NAMES.instanceRoleName,
      InstanceProfileName: NAMES.instanceProfileName,
    }),
  );
}),
new ScenarioOutput(
  "addedRoleToInstanceProfile",
  MESSAGES.addedRoleToInstanceProfile
    .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
),
...initParamsSteps,
new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
new ScenarioAction("createLaunchTemplate", async () => {
  const ssmClient = new SSMClient({});
  const { Parameter } = await ssmClient.send(
    new GetParameterCommand({
      Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",

```

```
    }),
  );
  const ec2Client = new EC2Client({});
  await ec2Client.send(
    new CreateLaunchTemplateCommand({
      LaunchTemplateName: NAMES.launchTemplateName,
      LaunchTemplateData: {
        InstanceType: "t3.micro",
        ImageId: Parameter.Value,
        IamInstanceProfile: { Name: NAMES.instanceProfileName },
        UserData: readFileSync(
          join(RESOURCES_PATH, "server_startup_script.sh"),
        ).toString("base64"),
        KeyName: NAMES.keyPairName,
      },
    }),
  );
}),
new ScenarioOutput(
  "createdLaunchTemplate",
  MESSAGES.createdLaunchTemplate.replace(
    "${LAUNCH_TEMPLATE_NAME}",
    NAMES.launchTemplateName,
  ),
),
new ScenarioOutput(
  "creatingAutoScalingGroup",
  MESSAGES.creatingAutoScalingGroup.replace(
    "${AUTO_SCALING_GROUP_NAME}",
    NAMES.autoScalingGroupName,
  ),
),
new ScenarioAction("createAutoScalingGroup", async (state) => {
  const ec2Client = new EC2Client({});
  const { AvailabilityZones } = await ec2Client.send(
    new DescribeAvailabilityZonesCommand({}),
  );
  state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
  const autoScalingClient = new AutoScalingClient({});
  await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
    autoScalingClient.send(
      new CreateAutoScalingGroupCommand({
        AvailabilityZones: state.availabilityZoneNames,
        AutoScalingGroupName: NAMES.autoScalingGroupName,
```

```

        LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
        },
        MinSize: 3,
        MaxSize: 3,
    )),
    ),
);
}),
new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
        MESSAGES.createdAutoScalingGroup
            .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
            .replace(
                "${AVAILABILITY_ZONE_NAMES}",
                state.availabilityZoneNames.join(", "),
            ),
    ),
new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
}),
new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
        new DescribeVpcsCommand({
            Filters: [{ Name: "is-default", Values: ["true"] }],
        }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
}),
new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
),
new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(

```

```
    new DescribeSubnetsCommand({
      Filters: [
        { Name: "vpc-id", Values: [state.defaultVpc] },
        { Name: "availability-zone", Values: state.availabilityZoneNames },
        { Name: "default-for-az", Values: ["true"] },
      ],
    }),
  );
  state.subnets = Subnets.map((subnet) => subnet.SubnetId);
}),
new ScenarioOutput(
  "gotSubnets",
  /**
   * @param {{ subnets: string[] }} state
   */
  (state) =>
    MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
),
new ScenarioOutput(
  "creatingLoadBalancerTargetGroup",
  MESSAGES.creatingLoadBalancerTargetGroup.replace(
    "${TARGET_GROUP_NAME}",
    NAMES.loadBalancerTargetGroupName,
  ),
),
new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new CreateTargetGroupCommand({
      Name: NAMES.loadBalancerTargetGroupName,
      Protocol: "HTTP",
      Port: 80,
      HealthCheckPath: "/healthcheck",
      HealthCheckIntervalSeconds: 10,
      HealthCheckTimeoutSeconds: 5,
      HealthyThresholdCount: 2,
      UnhealthyThresholdCount: 2,
      VpcId: state.defaultVpc,
    }),
  );
  const targetGroup = TargetGroups[0];
  state.targetGroupArn = targetGroup.TargetGroupArn;
  state.targetGroupProtocol = targetGroup.Protocol;
  state.targetGroupPort = targetGroup.Port;
```

```
    }),
    new ScenarioOutput(
      "createdLoadBalancerTargetGroup",
      MESSAGES.createdLoadBalancerTargetGroup.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      ),
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
```

```

        Port: state.targetGroupPort,
        DefaultActions: [
            { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
    })),
);
const listener = Listeners[0];
state.loadBalancerListenerArn = listener.ListenerArn;
}),
new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
        "${LB_LISTENER_ARN}",
        state.loadBalancerListenerArn,
    ),
),
new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
        .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
),
new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
        new AttachLoadBalancerTargetGroupsCommand({
            AutoScalingGroupName: NAMES.autoScalingGroupName,
            TargetGroupARNs: [state.targetGroupArn],
        })),
    );
}),
new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
),
new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-ec2').SecurityGroup}} state
     */
    async (state) => {
        const client = new EC2Client({});

```

```

const { SecurityGroups } = await client.send(
  new DescribeSecurityGroupsCommand({
    Filters: [{ Name: "group-name", Values: ["default"] }],
  }),
);
if (!SecurityGroups) {
  state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
}
state.defaultSecurityGroup = SecurityGroups[0];

/**
 * @type {string}
 */
const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
state.myIp = ipResponse.trim();
const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
  ({ IpRanges }) =>
    IpRanges.some(
      ({ CidrIp }) =>
        CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
    ),
)
  .filter(({ IpProtocol }) => IpProtocol === "tcp")
  .filter(({ FromPort }) => FromPort === 80);

state.myIpRules = myIpRules;
},
),
new ScenarioOutput(
  "verifiedInboundPort",
  /**
   * @param {{ myIpRules: any[] }} state
   */
  (state) => {
    if (state.myIpRules.length > 0) {
      return MESSAGES.foundIpRules.replace(
        "${IP_RULES}",
        JSON.stringify(state.myIpRules, null, 2),
      );
    }
    return MESSAGES.noIpRules;
  },
),
new ScenarioInput(

```



```
"shouldAddInboundRule",
/**
 * @param {{ myIpRules: any[] }} state
 */
(state) => {
  if (state.myIpRules.length > 0) {
    return false;
  }
  return MESSAGES.noIpRules;
},
{ type: "confirm" },
),
new ScenarioAction(
  "addInboundRule",
  /**
   * @param {{ defaultSecurityGroup: import('@aws-sdk/client-ec2').SecurityGroup }} state
   */
  async (state) => {
    if (!state.shouldAddInboundRule) {
      return;
    }

    const client = new EC2Client({});
    await client.send(
      new AuthorizeSecurityGroupIngressCommand({
        GroupId: state.defaultSecurityGroup.GroupId,
        CidrIp: `${state.myIp}/32`,
        FromPort: 80,
        ToPort: 80,
        IpProtocol: "tcp",
      })),
    );
  },
),
new ScenarioOutput("addedInboundRule", (state) => {
  if (state.shouldAddInboundRule) {
    return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
  }
  return false;
}),
new ScenarioOutput("verifyingEndpoint", (state) =>
  MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
),
```

```
new ScenarioAction("verifyEndpoint", async (state) => {
  try {
    const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
      axios.get(`http://${state.loadBalancerDns}`),
    );
    state.endpointResponse = JSON.stringify(response.data, null, 2);
  } catch (e) {
    state.verifyEndpointError = e;
  }
}),
new ScenarioOutput("verifiedEndpoint", (state) => {
  if (state.verifyEndpointError) {
    console.error(state.verifyEndpointError);
  } else {
    return MESSAGES.verifiedEndpoint.replace(
      "${ENDPOINT_RESPONSE}",
      state.endpointResponse,
    );
  }
}),
saveState,
];
```

Erstellen Sie Schritte, um die Demo auszuführen.

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
```

```
IAMClient,
CreatePolicyCommand,
CreateRoleCommand,
AttachRolePolicyCommand,
CreateInstanceProfileCommand,
AddRoleToInstanceProfileCommand,
waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
```

```

        throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
},
);

const getRecommendationResult = new ScenarioOutput(
    "getRecommendationResult",
    (state) =>
        `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
    { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
            Names: [NAMES.loadBalancerTargetGroupName],
        }),
    );
});

const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
        TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
);
state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
    "getHealthCheckResult",
    /**
     * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
    balancing-v2').TargetHealthDescription[]}} state
     */
    (state) => {
        const status = state.targetHealthDescriptions
            .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
            .join("\n");
        return `Health check:\n${status}`;
    },
    { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(

```

```
"loadBalancerLoop",
getRecommendation.action,
{
  whileConfig: {
    whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
    input: new ScenarioInput(
      "loadBalancerCheck",
      MESSAGES.demoLoadBalancerCheck,
      {
        type: "confirm",
      },
    ),
    output: getRecommendationResult,
  },
},
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[][]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
```

```
...statusSteps,
new ScenarioInput(
  "brokenDependencyConfirmation",
  MESSAGES.demoBrokenDependencyConfirmation,
  { type: "confirm" },
),
new ScenarioAction("brokenDependency", async (state) => {
  if (!state.brokenDependencyConfirmation) {
    process.exit();
  } else {
    const client = new SSMClient({});
    state.badTableName = `fake-table-${Date.now()}`;
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: state.badTableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }
}),
new ScenarioOutput("testBrokenDependency", (state) =>
  MESSAGES.demoTestBrokenDependency.replace(
    "${TABLE_NAME}",
    state.badTableName,
  ),
),
...statusSteps,
new ScenarioInput(
  "staticResponseConfirmation",
  MESSAGES.demoStaticResponseConfirmation,
  { type: "confirm" },
),
new ScenarioAction("staticResponse", async (state) => {
  if (!state.staticResponseConfirmation) {
    process.exit();
  } else {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmFailureResponseKey,
        Value: "static",
        Overwrite: true,
```

```

        Type: "String",
    })),
    );
}
}),
new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
...statusSteps,
new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
),
new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
        process.exit();
    }
}),
new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
        new PutParameterCommand({
            Name: NAMES.ssmTableNameKey,
            Value: NAMES.tableName,
            Overwrite: true,
            Type: "String",
        })),
    );
}),
new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
     */
    async (state) => {
        await createSsmOnlyInstanceProfile();
        const autoScalingClient = new AutoScalingClient({});
        const { AutoScalingGroups } = await autoScalingClient.send(
            new DescribeAutoScalingGroupsCommand({
                AutoScalingGroupNames: [NAMES.autoScalingGroupName],
            })),
        );
        state.targetInstance = AutoScalingGroups[0].Instances[0];
        const ec2Client = new EC2Client({});

```

```
const { IamInstanceProfileAssociations } = await ec2Client.send(
  new DescribeIamInstanceProfileAssociationsCommand({
    Filters: [
      { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
    ],
  }),
);
state.instanceProfileAssociationId =
  IamInstanceProfileAssociations[0].AssociationId;
await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
  ec2Client.send(
    new ReplaceIamInstanceProfileAssociationCommand({
      AssociationId: state.instanceProfileAssociationId,
      IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
    }),
  ),
);

await ec2Client.send(
  new RebootInstancesCommand({
    InstanceIds: [state.targetInstance.InstanceId],
  }),
);

const ssmClient = new SSMClient({});
await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
  const { InstanceInformationList } = await ssmClient.send(
    new DescribeInstanceInformationCommand({}),
  );

  const instance = InstanceInformationList.find(
    (info) => info.InstanceId === state.targetInstance.InstanceId,
  );

  if (!instance) {
    throw new Error("Instance not found.");
  }
});

await ssmClient.send(
  new SendCommandCommand({
    InstanceIds: [state.targetInstance.InstanceId],
    DocumentName: "AWS-RunShellScript",
    Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
  }),
);
```



```
    }),
  );
},
),
new ScenarioOutput(
  "testBadCredentials",
  /**
   * @param {{ targetInstance: import('@aws-sdk/client-
  ssm').InstanceInformation}} state
   */
  (state) =>
    MESSAGES.demoTestBadCredentials.replace(
      "${INSTANCE_ID}",
      state.targetInstance.InstanceId,
    ),
),
loadBalancerLoop,
new ScenarioInput(
  "deepHealthCheckConfirmation",
  MESSAGES.demoDeepHealthCheckConfirmation,
  { type: "confirm" },
),
new ScenarioAction("deepHealthCheckExit", (state) => {
  if (!state.deepHealthCheckConfirmation) {
    process.exit();
  }
}),
new ScenarioAction("deepHealthCheck", async () => {
  const client = new SSMClient({});
  await client.send(
    new PutParameterCommand({
      Name: NAMES.ssmHealthCheckKey,
      Value: "deep",
      Overwrite: true,
      Type: "String",
    }),
  ),
);
}),
new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
healthCheckLoop,
loadBalancerLoop,
new ScenarioInput(
  "killInstanceConfirmation",
  /**
```

```

    * @param {{ targetInstance: import('@aws-sdk/client-
    ssm').InstanceInformation }} state
    */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
      { type: "confirm" },
    ),
    new ScenarioAction("killInstanceExit", (state) => {
      if (!state.killInstanceConfirmation) {
        process.exit();
      }
    })),
    new ScenarioAction(
      "killInstance",
      /**
       * @param {{ targetInstance: import('@aws-sdk/client-
       ssm').InstanceInformation }} state
       */
      async (state) => {
        const client = new AutoScalingClient({});
        await client.send(
          new TerminateInstanceInAutoScalingGroupCommand({
            InstanceId: state.targetInstance.InstanceId,
            ShouldDecrementDesiredCapacity: false,
          })),
        );
      },
    ),
    new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
    healthCheckLoop,
    loadBalancerLoop,
    new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
      type: "confirm",
    })),
    new ScenarioAction("failOpenExit", (state) => {
      if (!state.failOpenConfirmation) {
        process.exit();
      }
    })),
    new ScenarioAction("failOpen", () => {
      const client = new SSMClient({});

```

```
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  })),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  })),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  })),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
```

```
        join(RESOURCES_PATH, "ssm_only_policy.json"),
    ),
  })),
);
await iamClient.send(
  new CreateRoleCommand({
    RoleName: NAMES.ssmOnlyRoleName,
    AssumeRolePolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Principal: { Service: "ec2.amazonaws.com" },
          Action: "sts:AssumeRole",
        },
      ],
    })),
  ));
await iamClient.send(
  new AttachRolePolicyCommand({
    RoleName: NAMES.ssmOnlyRoleName,
    PolicyArn: Policy.Arn,
  })),
);
await iamClient.send(
  new AttachRolePolicyCommand({
    RoleName: NAMES.ssmOnlyRoleName,
    PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
  })),
);
const { InstanceProfile } = await iamClient.send(
  new CreateInstanceProfileCommand({
    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
  })),
);
await waitUntilInstanceProfileExists(
  { client: iamClient },
  { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
);
await iamClient.send(
  new AddRoleToInstanceProfileCommand({
    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    RoleName: NAMES.ssmOnlyRoleName,
```

```
    }),  
  );  
  
  return InstanceProfile;  
}
```

Erstellen Sie Schritte, um alle Ressourcen zu vernichten.

```
import { unlinkSync } from "node:fs";  
  
import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";  
import {  
  EC2Client,  
  DeleteKeyPairCommand,  
  DeleteLaunchTemplateCommand,  
  RevokeSecurityGroupIngressCommand,  
} from "@aws-sdk/client-ec2";  
import {  
  IAMClient,  
  DeleteInstanceProfileCommand,  
  RemoveRoleFromInstanceProfileCommand,  
  DeletePolicyCommand,  
  DeleteRoleCommand,  
  DetachRolePolicyCommand,  
  paginateListPolicies,  
} from "@aws-sdk/client-iam";  
import {  
  AutoScalingClient,  
  DeleteAutoScalingGroupCommand,  
  TerminateInstanceInAutoScalingGroupCommand,  
  UpdateAutoScalingGroupCommand,  
  paginateDescribeAutoScalingGroups,  
} from "@aws-sdk/client-auto-scaling";  
import {  
  DeleteLoadBalancerCommand,  
  DeleteTargetGroupCommand,  
  DescribeTargetGroupsCommand,  
  ElasticLoadBalancingV2Client,  
} from "@aws-sdk/client-elastic-load-balancing-v2";  
  
import {  
  ScenarioOutput,
```

```
ScenarioInput,
ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    }
  )
];
```

```
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  })),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  })),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          })
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  })),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
  })
);
```

```
    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  })),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  })),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  })),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
```



```
    state.removeRoleFromInstanceProfileError = e;
  }
}),
new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
  if (state.removeRoleFromInstanceProfile) {
    console.error(state.removeRoleFromInstanceProfileError);
    return MESSAGES.removeRoleFromInstanceProfileError
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }
  return MESSAGES.removedRoleFromInstanceProfile
    .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
}),
new ScenarioAction("deleteInstanceRole", async (state) => {
  try {
    const client = new IAMClient({});
    await client.send(
      new DeleteRoleCommand({
        RoleName: NAMES.instanceRoleName,
      }),
    );
  } catch (e) {
    state.deleteInstanceRoleError = e;
  }
}),
new ScenarioOutput("deleteInstanceRoleResult", (state) => {
  if (state.deleteInstanceRoleError) {
    console.error(state.deleteInstanceRoleError);
    return MESSAGES.deleteInstanceRoleError.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }
  return MESSAGES.deletedInstanceRole.replace(
    "${INSTANCE_ROLE_NAME}",
    NAMES.instanceRoleName,
  );
}),
new ScenarioAction("deleteInstanceProfile", async (state) => {
  try {
    const client = new IAMClient({});
    await client.send(
      new DeleteInstanceProfileCommand({
```

```
        InstanceProfileName: NAMES.instanceProfileName,
      )),
    );
  } catch (e) {
    state.deleteInstanceProfileError = e;
  }
  )),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  )),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  )),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  )),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
```

```
const client = new EC2Client({});
try {
  await client.send(
    new DeleteLaunchTemplateCommand({
      LaunchTemplateName: NAMES.launchTemplateName,
    }),
  );
} catch (e) {
  state.deleteLaunchTemplateError = e;
}
}),
new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
  if (state.deleteLaunchTemplateError) {
    console.error(state.deleteLaunchTemplateError);
    return MESSAGES.deleteLaunchTemplateError.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }
  return MESSAGES.deletedLaunchTemplate.replace(
    "${LAUNCH_TEMPLATE_NAME}",
    NAMES.launchTemplateName,
  );
}),
new ScenarioAction("deleteLoadBalancer", async (state) => {
  try {
    const client = new ElasticLoadBalancingV2Client({});
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    await client.send(
      new DeleteLoadBalancerCommand({
        LoadBalancerArn: loadBalancer.LoadBalancerArn,
      }),
    );
    await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
      const lb = await findLoadBalancer(NAMES.loadBalancerName);
      if (lb) {
        throw new Error("Load balancer still exists.");
      }
    });
  } catch (e) {
    state.deleteLoadBalancerError = e;
  }
}),
new ScenarioOutput("deleteLoadBalancerResult", (state) => {
```

```
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  })),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  })),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  });
```

```
    })),
    new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
      try {
        const client = new IAMClient({});
        await client.send(
          new RemoveRoleFromInstanceProfileCommand({
            InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
            RoleName: NAMES.ssmOnlyRoleName,
          })),
        );
      } catch (e) {
        state.detachSsmOnlyRoleFromProfileError = e;
      }
    })),
    new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
      if (state.detachSsmOnlyRoleFromProfileError) {
        console.error(state.detachSsmOnlyRoleFromProfileError);
        return MESSAGES.detachSsmOnlyRoleFromProfileError
          .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
          .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
      }
      return MESSAGES.detachedSsmOnlyRoleFromProfile
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    })),
    new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
      try {
        const iamClient = new IAMClient({});
        const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
        await iamClient.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.ssmOnlyRoleName,
            PolicyArn: ssmOnlyPolicy.Arn,
          })),
        );
      } catch (e) {
        state.detachSsmOnlyCustomRolePolicyError = e;
      }
    })),
    new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
      if (state.detachSsmOnlyCustomRolePolicyError) {
        console.error(state.detachSsmOnlyCustomRolePolicyError);
        return MESSAGES.detachSsmOnlyCustomRolePolicyError
          .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      }
    })
  ]
}
```

```
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    })),
    new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
        try {
            const iamClient = new IAMClient({});
            await iamClient.send(
                new DetachRolePolicyCommand({
                    RoleName: NAMES.ssmOnlyRoleName,
                    PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
                }),
            );
        } catch (e) {
            state.detachSsmOnlyAWSRolePolicyError = e;
        }
    })),
    new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
        if (state.detachSsmOnlyAWSRolePolicyError) {
            console.error(state.detachSsmOnlyAWSRolePolicyError);
            return MESSAGES.detachSsmOnlyAWSRolePolicyError
                .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
                .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
        }
        return MESSAGES.detachedSsmOnlyAWSRolePolicy
            .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
            .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    })),
    new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
        try {
            const iamClient = new IAMClient({});
            await iamClient.send(
                new DeleteInstanceProfileCommand({
                    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
                }),
            );
        } catch (e) {
            state.deleteSsmOnlyInstanceProfileError = e;
        }
    })),
    new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
        if (state.deleteSsmOnlyInstanceProfileError) {
```

```
    console.error(state.deleteSsmOnlyInstanceProfileError);
    return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }
  return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
    "${INSTANCE_PROFILE_NAME}",
    NAMES.ssmOnlyInstanceProfileName,
  );
}),
new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
  try {
    const iamClient = new IAMClient({});
    const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
    await iamClient.send(
      new DeletePolicyCommand({
        PolicyArn: ssmOnlyPolicy.Arn,
      }),
    );
  } catch (e) {
    state.deleteSsmOnlyPolicyError = e;
  }
}),
new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
  if (state.deleteSsmOnlyPolicyError) {
    console.error(state.deleteSsmOnlyPolicyError);
    return MESSAGES.deleteSsmOnlyPolicyError.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }
  return MESSAGES.deletedSsmOnlyPolicy.replace(
    "${POLICY_NAME}",
    NAMES.ssmOnlyPolicyName,
  );
}),
new ScenarioAction("deleteSsmOnlyRole", async (state) => {
  try {
    const iamClient = new IAMClient({});
    await iamClient.send(
      new DeleteRoleCommand({
        RoleName: NAMES.ssmOnlyRoleName,
      }),
    );
  }
}),
```

```
    );
  } catch (e) {
    state.deleteSsmOnlyRoleError = e;
  }
}),
new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
  if (state.deleteSsmOnlyRoleError) {
    console.error(state.deleteSsmOnlyRoleError);
    return MESSAGES.deleteSsmOnlyRoleError.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }
  return MESSAGES.deletedSsmOnlyRole.replace(
    "${ROLE_NAME}",
    NAMES.ssmOnlyRoleName,
  );
}),
new ScenarioAction(
  "revokeSecurityGroupIngress",
  async (
    /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
    state,
  ) => {
    const ec2Client = new EC2Client({});

    try {
      await ec2Client.send(
        new RevokeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        })),
    );
  } catch (e) {
    state.revokeSecurityGroupIngressError = e;
  }
},
),
new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
  if (state.revokeSecurityGroupIngressError) {
    console.error(state.revokeSecurityGroupIngressError);
  }
}),

```



```
        return MESSAGES.revokeSecurityGroupIngressError.replace(
            "${IP}",
            state.myIp,
        );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
}),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
    const client = new IAMClient({});
    const paginatedPolicies = paginateListPolicies({ client }, {});
    for await (const page of paginatedPolicies) {
        const policy = page.Policies.find((p) => p.PolicyName === policyName);
        if (policy) {
            return policy;
        }
    }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
    const client = new AutoScalingClient({});
    try {
        await client.send(
            new DeleteAutoScalingGroupCommand({
                AutoScalingGroupName: groupName,
            }),
        );
    } catch (err) {
        if (!(err instanceof Error)) {
            throw err;
        }
        console.log(err.name);
        throw err;
    }
}

/**
```

```
* @param {string} groupName
*/
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- API-Einheiten finden Sie unter den folgenden Themen in der AWS SDK for JavaScript API-Referenz.
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)

- [CreateInstanceProfile](#)
- [CreateLaunchTemplate](#)
- [CreateListener](#)
- [CreateLoadBalancer](#)
- [CreateTargetGroup](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInstanceProfile](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplacelamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Sie sehen das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-Repository](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario an einer Eingabeaufforderung aus.

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
    resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example,
        such as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
        DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
        """
```

```
"""
recommendations_path = f"{self.resource_path}/recommendations.json"
startup_script = f"{self.resource_path}/server_startup_script.sh"
instance_policy = f"{self.resource_path}/instance_policy.json"

logging.info("Starting deployment of resources for the resilient
service.")

logging.info(
    "Creating and populating DynamoDB table '%s'.",
    self.recommendation.table_name,
)
self.recommendation.create()
self.recommendation.populate(recommendations_path)

logging.info(
    "Creating an EC2 launch template with the startup script '%s'.",
    startup_script,
)
self.autoscaler.create_template(startup_script, instance_policy)

logging.info(
    "Creating an EC2 Auto Scaling group across multiple Availability
Zones."
)
zones = self.autoscaler.create_autoscaling_group(3)

logging.info("Creating variables that control the flow of the demo.")
self.param_helper.reset()

logging.info("Creating Elastic Load Balancing target group and load
balancer.")

vpc = self.autoscaler.get_default_vpc()
subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
target_group = self.loadbalancer.create_target_group(
    self.target_group_name, self.protocol, self.port, vpc["VpcId"]
)
self.loadbalancer.create_load_balancer(
    self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
)
self.loadbalancer.create_listener(self.load_balancer_name, target_group)

self.autoscaler.attach_load_balancer_target_group(target_group)
```

```
logging.info("Verifying access to the load balancer endpoint.")
endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

if not lb_success:
    logging.warning(
        "Couldn't connect to the load balancer. Verifying that the port
is open..."
    )
    sec_group, port_is_open = self.autoscaler.verify_inbound_port(
        vpc, self.port, current_ip_address
    )
    sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
        vpc, self.ssh_port, current_ip_address
    )
    if not port_is_open:
        logging.warning(
            "The default security group for your VPC must allow access
from this computer."
        )
        if q.ask(
            f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
            f"inbound traffic on port {self.port} from your computer's IP
address of {current_ip_address}? (y/n) ",
            q.is_yesno,
        ):
            self.autoscaler.open_inbound_port(
                sec_group["GroupId"], self.port, current_ip_address
            )
    if not ssh_port_is_open:
        if q.ask(
            f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
            f"inbound SSH traffic on port {self.ssh_port} for debugging
from your computer's IP address of {current_ip_address}? (y/n) ",
            q.is_yesno,
        ):
            self.autoscaler.open_inbound_port(
                sec_group["GroupId"], self.ssh_port, current_ip_address
            )
```

```
        lb_success =
self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer
endpoint. Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as
sending requests to
the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer
endpoint.")
                endpoint =
self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
```

```
        state = target["TargetHealth"]["State"]
        logging.info(
            "Target %s on port %d is %s",
            target["Target"]["Id"],
            target["Target"]["Port"],
            state,
        )
        if state != "healthy":
            logging.warning(
                "%s: %s",
                target["TargetHealth"]["Reason"],
                target["TargetHealth"]["Description"],
            )
        logging.info(
            "Note that it can take a minute or two for the health check
to update."
        )
    elif choice == 2:
        logging.info("Proceeding to the next part of the demo.")

def demo(self) -> None:
    """
    Runs the demonstration, showing how the service responds to different
failure scenarios
and how a resilient architecture can keep the service running.
    """
    ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

    logging.info("Resetting parameters to starting values for the demo.")
    self.param_helper.reset()

    logging.info(
        "Starting demonstration of the service's resilience under various
failure conditions."
    )
    self.demo_choices()

    logging.info(
        "Simulating failure by changing the Systems Manager parameter to a
non-existent table."
    )
    self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
    logging.info("Sending GET requests will now return failure codes.")
    self.demo_choices()
```



```
logging.info("Switching to static response mode to mitigate failure.")
self.param_helper.put(self.param_helper.failure_response, "static")
logging.info("Sending GET requests will now return static responses.")
self.demo_choices()

logging.info("Restoring normal operation of the recommendation service.")
self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

logging.info(
    "Introducing a failure by assigning bad credentials to one of the
instances."
)
self.autoscaler.create_instance_profile(
    ssm_only_policy,
    self.autoscaler.bad_creds_policy_name,
    self.autoscaler.bad_creds_role_name,
    self.autoscaler.bad_creds_profile_name,
    ["AmazonSSMManagedInstanceCore"],
)
instances = self.autoscaler.get_instances()
bad_instance_id = instances[0]
instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
logging.info(
    "Replacing instance profile with bad credentials for instance %s.",
    bad_instance_id,
)
self.autoscaler.replace_instance_profile(
    bad_instance_id,
    self.autoscaler.bad_creds_profile_name,
    instance_profile["AssociationId"],
)
logging.info(
    "Sending GET requests may return either a valid recommendation or a
static response."
)
self.demo_choices()

logging.info("Implementing deep health checks to detect unhealthy
instances.")
self.param_helper.put(self.param_helper.health_check, "deep")
logging.info("Checking the health of the load balancer targets.")
self.demo_choices()
```

```
        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace
it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance
replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation
service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
        if automation:
            cleanup = True
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
```

```
        self.autoscaler.delete_instance_profile(
            self.autoscaler.bad_creds_profile_name,
            self.autoscaler.bad_creds_role_name,
        )
        logging.info("Deleting DynamoDB table and other resources.")
        self.recommendation.destroy()
    else:
        logging.warning(
            "Resources have not been deleted. Ensure you clean them up
            manually to avoid unexpected charges."
        )

def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the
    demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
        are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../../workflows/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM
        policies and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
```

```
ec2_client = boto3.client("ec2")
ssm_client = boto3.client("ssm")
iam_client = boto3.client("iam")

# Wrapper instantiations
recommendation = RecommendationService(
    "doc-example-recommendation-service", ddb_client
)
autoscaling_wrapper = AutoScalingWrapper(
    prefix,
    "t3.micro",
    "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
    autoscaling_client,
    ec2_client,
    ssm_client,
    iam_client,
)
elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
param_helper = ParameterHelper(recommendation.table_name, ssm_client)

# Demo invocation
runner = Runner(
    args.resource_path,
    recommendation,
    autoscaling_wrapper,
    elb_wrapper,
    param_helper,
)
actions = [args.action] if args.action != "all" else ["deploy", "demo",
"destroy"]
for action in actions:
    if action == "deploy":
        runner.deploy()
    elif action == "demo":
        runner.demo()
    elif action == "destroy":
        runner.destroy()

logging.info("Demo completed successfully.")

if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

Erstellen Sie eine Klasse, die Auto Scaling- und EC2 Amazon-Aktionen umschließt.

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
```

```
self.group_name = f"{resource_prefix}-group"

# Happy path
self.instance_policy_name = f"{resource_prefix}-pol"
self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"

# Failure mode
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

def create_policy(self, policy_file: str, policy_name: str) -> str:
    """
    Creates a new IAM policy or retrieves the ARN of an existing policy.

    :param policy_file: The path to a JSON file that contains the policy
    definition.
    :param policy_name: The name to give the created policy.
    :return: The ARN of the created or existing policy.
    """
    with open(policy_file) as file:
        policy_doc = file.read()

    try:
        response = self.iam_client.create_policy(
            PolicyName=policy_name, PolicyDocument=policy_doc
        )
        policy_arn = response["Policy"]["Arn"]
        log.info(f"Policy '{policy_name}' created successfully. ARN:
{policy_arn}")
        return policy_arn

    except ClientError as err:
        if err.response["Error"]["Code"] == "EntityAlreadyExists":
            # If the policy already exists, get its ARN
            response = self.iam_client.get_policy(
                PolicyArn=f"arn:aws:iam::{self.account_id}:policy/
{policy_name}"
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' already exists. ARN:
{policy_arn}")
```

```

        return policy_arn
        log.error(f"Full error:\n\t{err}")

def create_role(self, role_name: str, assume_role_doc: dict) -> str:
    """
    Creates a new IAM role or retrieves the ARN of an existing role.

    :param role_name: The name to give the created role.
    :param assume_role_doc: The assume role policy document that specifies
which
                           entities can assume the role.
    :return: The ARN of the created or existing role.
    """
    try:
        response = self.iam_client.create_role(
            RoleName=role_name,
AssumeRolePolicyDocument=json.dumps(assume_role_doc)
        )
        role_arn = response["Role"]["Arn"]
        log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
        return role_arn

    except ClientError as err:
        if err.response["Error"]["Code"] == "EntityAlreadyExists":
            # If the role already exists, get its ARN
            response = self.iam_client.get_role(RoleName=role_name)
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
            return role_arn
        log.error(f"Full error:\n\t{err}")

def attach_policy(
    self,
    role_name: str,
    policy_arn: str,
    aws_managed_policies: Tuple[str, ...] = (),
) -> None:
    """
    Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

    :param role_name: The name of the role to attach the policy to.
    :param policy_arn: The ARN of the policy to attach.

```

```

        :param aws_managed_policies: A tuple of AWS-managed policy names to
attach to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role
{role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is
assumed by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
            the role, such as
AmazonSSMManagedInstanceCore to grant

```


use of Systems Manager to send commands to the instance.

```

:return: The ARN of the profile that is created.
"""
assume_role_doc = {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {"Service": "ec2.amazonaws.com"},
            "Action": "sts:AssumeRole",
        }
    ],
}
policy_arn = self.create_policy(policy_file, policy_name)
self.create_role(role_name, assume_role_doc)
self.attach_policy(role_name, policy_arn, aws_managed_policies)

try:
    profile_response = self.iam_client.create_instance_profile(
        InstanceProfileName=profile_name
    )
    waiter = self.iam_client.get_waiter("instance_profile_exists")
    waiter.wait(InstanceProfileName=profile_name)
    time.sleep(10) # wait a little longer
    profile_arn = profile_response["InstanceProfile"]["Arn"]
    self.iam_client.add_role_to_instance_profile(
        InstanceProfileName=profile_name, RoleName=role_name
    )
    log.info("Created profile %s and added role %s.", profile_name,
role_name)
except ClientError as err:
    if err.response["Error"]["Code"] == "EntityAlreadyExists":
        prof_response = self.iam_client.get_instance_profile(
            InstanceProfileName=profile_name
        )
        profile_arn = prof_response["InstanceProfile"]["Arn"]
        log.info(
            "Instance profile %s already exists, nothing to do.",
profile_name
        )
        log.error(f"Full error:\n\t{err}")
    return profile_arn

```

```
def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
    """
    Gets data about the profile associated with an instance.

    :param instance_id: The ID of the instance to look up.
    :return: The profile data.
    """
    try:
        response =
self.ec2_client.describe_iam_instance_profile_associations(
            Filters=[{"Name": "instance-id", "Values": [instance_id]}]
        )
        if not response["IamInstanceProfileAssociations"]:
            log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
{instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
                log.error(f"Full error:\n\t{err}")

def replace_instance_profile(
    self,
    instance_id: str,
    new_instance_profile_name: str,
    profile_association_id: str,
) -> None:
    """
    Replaces the profile associated with a running instance. After the
profile is
    replaced, the instance is rebooted to ensure that it uses the new
profile. When
    the instance is ready, Systems Manager is used to restart the Python web
server.
```

```
        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                instance.
"""
try:
    self.ec2_client.replace_iam_instance_profile_association(
        IamInstanceProfile={"Name": new_instance_profile_name},
        AssociationId=profile_association_id,
    )
    log.info(
        "Replaced instance profile for association %s with profile %s.",
        profile_association_id,
        new_instance_profile_name,
    )
    time.sleep(5)

    self.ec2_client.reboot_instances(InstanceIds=[instance_id])
    log.info("Rebooting instance %s.", instance_id)
    waiter = self.ec2_client.get_waiter("instance_running")
    log.info("Waiting for instance %s to be running.", instance_id)
    waiter.wait(InstanceIds=[instance_id])
    log.info("Instance %s is now running.", instance_id)

    self.ssm_client.send_command(
        InstanceIds=[instance_id],
        DocumentName="AWS-RunShellScript",
        Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
    )
    log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
except ClientError as err:
    log.error("Failed to replace instance profile.")
    error_code = err.response["Error"]["Code"]
    if error_code == "InvalidAssociationID.NotFound":
        log.error(
            f"Association ID '{profile_association_id}' does not exist."
            "Please check the association ID and try again."
        )
    if error_code == "InvalidInstanceId":
        log.error(
```

```
        f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
        f"Please verify the instance ID and try again."
    )
    log.error(f"Full error:\n\t{err}")

def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
    """
    Detaches a role from an instance profile, detaches policies from the
role,
and deletes all the resources.

:param profile_name: The name of the profile to delete.
:param role_name: The name of the role to delete.
    """
    try:
        self.iam_client.remove_role_from_instance_profile(
            InstanceProfileName=profile_name, RoleName=role_name
        )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
        log.info("Deleted instance profile %s.", profile_name)
        attached_policies = self.iam_client.list_attached_role_policies(
            RoleName=role_name
        )
        for pol in attached_policies["AttachedPolicies"]:
            self.iam_client.detach_role_policy(
                RoleName=role_name, PolicyArn=pol["PolicyArn"]
            )
            if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
    except ClientError as err:
        log.error(
            f"Couldn't delete instance profile {profile_name} or detach "
            f"policies and delete role {role_name}: {err}"
        )
        if err.response["Error"]["Code"] == "NoSuchEntity":
            log.info(
                "Instance profile %s doesn't exist, nothing to do.",
profile_name
```

```
    )

def create_key_pair(self, key_pair_name: str) -> None:
    """
    Creates a new key pair.

    :param key_pair_name: The name of the key pair to create.
    """
    try:
        response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
        with open(f"{key_pair_name}.pem", "w") as file:
            file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(f"Failed to create key pair {key_pair_name}.")
        if error_code == "InvalidKeyPair.Duplicate":
            log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
        log.error(f"Full error:\n\t{err}")

def delete_key_pair(self) -> None:
    """
    Deletes a key pair.
    """
    try:
        self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
        remove(f"{self.key_pair_name}.pem")
        log.info("Deleted key pair %s.", self.key_pair_name)
    except ClientError as err:
        log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
        log.error(f"Full error:\n\t{err}")
    except FileNotFoundError as err:
        log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
        log.error(f"Full error:\n\t{err}")

def create_template(
    self, server_startup_script_file: str, instance_policy_file: str
) -> Dict[str, Any]:
```

```
"""
    Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
    launch template specifies a Bash script in its user data field that runs
after
    the instance is started. This script installs Python packages and starts
a
    Python web server on the instance.

    :param server_startup_script_file: The path to a Bash script file that is
run
                                     when an instance starts.
    :param instance_policy_file: The path to a file that defines a
permissions policy
                                to create and attach to the instance
profile.
    :return: Information about the newly created template.
"""
template = {}
try:
    # Create key pair and instance profile
    self.create_key_pair(self.key_pair_name)
    self.create_instance_profile(
        instance_policy_file,
        self.instance_policy_name,
        self.instance_role_name,
        self.instance_profile_name,
    )

    # Read the startup script
    with open(server_startup_script_file) as file:
        start_server_script = file.read()

    # Get the latest AMI ID
    ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
    ami_id = ami_latest["Parameter"]["Value"]

    # Create the launch template
    lt_response = self.ec2_client.create_launch_template(
        LaunchTemplateName=self.launch_template_name,
        LaunchTemplateData={
            "InstanceType": self.inst_type,
            "ImageId": ami_id,
            "IamInstanceProfile": {"Name": self.instance_profile_name},
```

```
        "UserData": base64.b64encode(
            start_server_script.encode(encoding="utf-8")
        ).decode(encoding="utf-8"),
        "KeyName": self.key_pair_name,
    },
)
template = lt_response["LaunchTemplate"]
log.info(
    f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
)
except ClientError as err:
    log.error(f"Failed to create launch template
{self.launch_template_name}.")
    error_code = err.response["Error"]["Code"]
    if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
        log.info(
            f"Launch template {self.launch_template_name} already exists,
nothing to do."
        )
    log.error(f"Full error:\n\t{err}")
return template

def delete_template(self):
    """
    Deletes a launch template.
    """
    try:
        self.ec2_client.delete_launch_template(
            LaunchTemplateName=self.launch_template_name
        )
        self.delete_instance_profile(
            self.instance_profile_name, self.instance_role_name
        )
        log.info("Launch template %s deleted.", self.launch_template_name)
    except ClientError as err:
        if (
            err.response["Error"]["Code"]
            == "InvalidLaunchTemplateName.NotFoundException"
        ):
            log.info(
                "Launch template %s does not exist, nothing to do.",
                self.launch_template_name,
```

```
        )
        log.error(f"Full error:\n\t{err}")

def get_availability_zones(self) -> List[str]:
    """
    Gets a list of Availability Zones in the AWS Region of the Amazon EC2
    client.

    :return: The list of Availability Zones for the client Region.
    """
    try:
        response = self.ec2_client.describe_availability_zones()
        zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
        log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
    except ClientError as err:
        log.error("Failed to retrieve availability zones.")
        log.error(f"Full error:\n\t{err}")
    else:
        return zones

def create_autoscaling_group(self, group_size: int) -> List[str]:
    """
    Creates an EC2 Auto Scaling group with the specified size.

    :param group_size: The number of instances to set for the minimum and
    maximum in
                        the group.
    :return: The list of Availability Zones specified for the group.
    """
    try:
        zones = self.get_availability_zones()
        self.autoscaling_client.create_auto_scaling_group(
            AutoScalingGroupName=self.group_name,
            AvailabilityZones=zones,
            LaunchTemplate={
                "LaunchTemplateName": self.launch_template_name,
                "Version": "$Default",
            },
            MinSize=group_size,
            MaxSize=group_size,
        )
        log.info(
```



```
        f"Created EC2 Auto Scaling group {self.group_name} with
availability zones {zones}."
    )
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        if error_code == "AlreadyExists":
            log.info(
                f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
            )
        else:
            log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
            log.error(f"Full error:\n\t{err}")
    else:
        return zones

def get_instances(self) -> List[str]:
    """
    Gets data about the instances in the EC2 Auto Scaling group.

    :return: A list of instance IDs in the Auto Scaling group.
    """
    try:
        as_response = self.autoscaling_client.describe_auto_scaling_groups(
            AutoScalingGroupNames=[self.group_name]
        )
        instance_ids = [
            i["InstanceId"]
            for i in as_response["AutoScalingGroups"][0]["Instances"]
        ]
        log.info(
            f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
        )
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(
            f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
        )
        if error_code == "ResourceNotFound":
```

```
        log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
        log.error(f"Full error:\n\t{err}")
    else:
        return instance_ids

def terminate_instance(self, instance_id: str, decrementssetting=False) ->
None:
    """
    Terminates an instance in an EC2 Auto Scaling group. After an instance is
    terminated, it can no longer be accessed.

    :param instance_id: The ID of the instance to terminate.
    :param decrementssetting: If True, do not replace terminated instances.
    """
    try:
        self.autoscaling_client.terminate_instance_in_auto_scaling_group(
            InstanceId=instance_id,
            ShouldDecrementDesiredCapacity=decrementssetting,
        )
        log.info("Terminated instance %s.", instance_id)

        # Adding a waiter to ensure the instance is terminated
        waiter = self.ec2_client.get_waiter("instance_terminated")
        log.info("Waiting for instance %s to be terminated...", instance_id)
        waiter.wait(InstanceIds=[instance_id])
        log.info(
            f"Instance '{instance_id}' has been terminated and will be
replaced."
        )

    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(f"Failed to terminate instance '{instance_id}'.")
        if error_code == "ScalingActivityInProgressFault":
            log.error(
                "Scaling activity is currently in progress. "
                "Wait for the scaling activity to complete before attempting
to terminate the instance again."
            )
        elif error_code == "ResourceContentionFault":
            log.error(
                "The request failed due to a resource contention issue. "
```

```
        "Ensure that no conflicting operations are being performed on
the resource."
    )
    log.error(f"Full error:\n\t{err}")

def attach_load_balancer_target_group(
    self, lb_target_group: Dict[str, Any]
) -> None:
    """
    Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
    The target group specifies how the load balancer forwards requests to the
instances
    in the group.

    :param lb_target_group: Data about the ELB target group to attach.
    """
    try:
        self.autoscaling_client.attach_load_balancer_target_groups(
            AutoScalingGroupName=self.group_name,
            TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
        )
        log.info(
            "Attached load balancer target group %s to auto scaling group
%s.",
            lb_target_group["TargetGroupName"],
            self.group_name,
        )
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(
            f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
        )
        if error_code == "ResourceContentionFault":
            log.error(
                "The request failed due to a resource contention issue. "
                "Ensure that no conflicting operations are being performed on
the resource."
            )
        elif error_code == "ServiceLinkedRoleFailure":
            log.error(
                "The operation failed because the service-linked role is not
ready or does not exist. "
```

```

        "Check that the service-linked role exists and is correctly
configured."
    )
    log.error(f"Full error:\n\t{err}")

def delete_autoscaling_group(self, group_name: str) -> None:
    """
    Terminates all instances in the group, then deletes the EC2 Auto Scaling
group.

:param group_name: The name of the group to delete.
    """
    try:
        response = self.autoscaling_client.describe_auto_scaling_groups(
            AutoScalingGroupNames=[group_name]
        )
        groups = response.get("AutoScalingGroups", [])
        if len(groups) > 0:
            self.autoscaling_client.update_auto_scaling_group(
                AutoScalingGroupName=group_name, MinSize=0
            )
            instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
            for inst_id in instance_ids:
                self.terminate_instance(inst_id)

            # Wait for all instances to be terminated
            if instance_ids:
                waiter = self.ec2_client.get_waiter("instance_terminated")
                log.info("Waiting for all instances to be terminated...")
                waiter.wait(InstanceIds=instance_ids)
                log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
        if error_code == "ScalingActivityInProgressFault":
            log.error(
                "Scaling activity is currently in progress. "
                "Wait for the scaling activity to complete before attempting
to delete the group again."
            )

```

```
        elif error_code == "ResourceContentionFault":
            log.error(
                "The request failed due to a resource contention issue. "
                "Ensure that no conflicting operations are being performed on
the group."
            )
            log.error(f"Full error:\n\t{err}")

def get_default_vpc(self) -> Dict[str, Any]:
    """
    Gets the default VPC for the account.

    :return: Data about the default VPC.
    """
    try:
        response = self.ec2_client.describe_vpcs(
            Filters=[{"Name": "is-default", "Values": ["true"]}])
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error("Failed to retrieve the default VPC.")
        if error_code == "UnauthorizedOperation":
            log.error(
                "You do not have the necessary permissions to describe VPCs.
"
                "Ensure that your AWS IAM user or role has the correct
permissions."
            )
        elif error_code == "InvalidParameterValue":
            log.error(
                "One or more parameters are invalid. Check the request
parameters."
            )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass
```

```
def verify_inbound_port(
    self, vpc: Dict[str, Any], port: int, ip_address: str
) -> Tuple[Dict[str, Any], bool]:
    """
    Verify the default security group of the specified VPC allows ingress
    from this
    computer. This can be done by allowing ingress from this computer's IP
    address. In some situations, such as connecting from a corporate network,
    you
    must instead specify a prefix list ID. You can also temporarily open the
    port to
    any IP address while running this example. If you do, be sure to remove
    public
    access when you're done.

    :param vpc: The VPC used by this example.
    :param port: The port to verify.
    :param ip_address: This computer's IP address.
    :return: The default security group of the specified VPC, and a value
    that indicates
           whether the specified port is open.
    """
    try:
        response = self.ec2_client.describe_security_groups(
            Filters=[
                {"Name": "group-name", "Values": ["default"]},
                {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
            ]
        )
        sec_group = response["SecurityGroups"][0]
        port_is_open = False
        log.info(f"Found default security group {sec_group['GroupId']}.")

        for ip_perm in sec_group["IpPermissions"]:
            if ip_perm.get("FromPort", 0) == port:
                log.info(f"Found inbound rule: {ip_perm}")
                for ip_range in ip_perm["IpRanges"]:
                    cidr = ip_range.get("CidrIp", "")
                    if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                        port_is_open = True
                if ip_perm["PrefixListIds"]:
                    port_is_open = True
        if not port_is_open:
```

```

        log.info(
            f"The inbound rule does not appear to be open to
either this computer's IP "
            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
        )
    else:
        break
except ClientError as err:
    error_code = err.response["Error"]["Code"]
    log.error(
        f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
    )
    if error_code == "InvalidVpcID.NotFound":
        log.error(
            f"The specified VPC ID '{vpc['VpcId']}' does not exist.
Please check the VPC ID."
        )
        log.error(f"Full error:\n\t{err}")
    else:
        return sec_group, port_is_open

def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
    """
    Add an ingress rule to the specified security group that allows access on
the
specified port from the specified IP address.

:param sec_group_id: The ID of the security group to modify.
:param port: The port to open.
:param ip_address: The IP address that is granted access.
    """
    try:
        self.ec2_client.authorize_security_group_ingress(
            GroupId=sec_group_id,
            CidrIp=f"{ip_address}/32",
            FromPort=port,
            ToPort=port,
            IpProtocol="tcp",
        )
        log.info(

```

```

        "Authorized ingress to %s on port %s from %s.",
        sec_group_id,
        port,
        ip_address,
    )
except ClientError as err:
    error_code = err.response["Error"]["Code"]
    log.error(
        f"Failed to authorize ingress to security group '{sec_group_id}'
on port {port} from {ip_address}."
    )
    if error_code == "InvalidGroupId.Malformed":
        log.error(
            "The security group ID is malformed. "
            "Please verify that the security group ID is correct."
        )
    elif error_code == "InvalidPermission.Duplicate":
        log.error(
            "The specified rule already exists in the security group. "
            "Check the existing rules for this security group."
        )
    log.error(f"Full error:\n\t{err}")

def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
    """
    Gets the default subnets in a VPC for a specified list of Availability
    Zones.

    :param vpc_id: The ID of the VPC to look up.
    :param zones: The list of Availability Zones to look up.
    :return: The list of subnets found.
    """
    # Ensure that 'zones' is a list, even if None is passed
    if zones is None:
        zones = []
    try:
        paginator = self.ec2_client.get_paginator("describe_subnets")
        page_iterator = paginator.paginate(
            Filters=[
                {"Name": "vpc-id", "Values": [vpc_id]},
                {"Name": "availability-zone", "Values": zones},
                {"Name": "default-for-az", "Values": ["true"]},
            ],

```



```

    ]
)

subnets = []
for page in page_iterator:
    subnets.extend(page["Subnets"])

log.info("Found %s subnets for the specified zones.", len(subnets))
return subnets
except ClientError as err:
    log.error(
        f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
{zones}."
    )
    error_code = err.response["Error"]["Code"]
    if error_code == "InvalidVpcID.NotFound":
        log.error(
            "The specified VPC ID does not exist. "
            "Please check the VPC ID and try again."
        )
    # Add more error-specific handling as needed
    log.error(f"Full error:\n\t{err}")

```

Erstellen Sie eine Klasse, die Elastic-Load-Balancing-Aktionen beinhaltet.

```

class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client

    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """

```

Creates an Elastic Load Balancing target group. The target group specifies how the load balancer forwards requests to instances in the group and how instance health is checked.

To speed up this demo, the health check is configured with shortened times and lower thresholds. In production, you might want to decrease the sensitivity of your health checks to avoid unwanted failures.

```
:param target_group_name: The name of the target group to create.
:param protocol: The protocol to use to forward requests, such as 'HTTP'.
:param port: The port to use to forward requests, such as 80.
:param vpc_id: The ID of the VPC in which the load balancer exists.
:return: Data about the newly created target group.
"""
try:
    response = self.elb_client.create_target_group(
        Name=target_group_name,
        Protocol=protocol,
        Port=port,
        HealthCheckPath="/healthcheck",
        HealthCheckIntervalSeconds=10,
        HealthCheckTimeoutSeconds=5,
        HealthyThresholdCount=2,
        UnhealthyThresholdCount=2,
        VpcId=vpc_id,
    )
    target_group = response["TargetGroups"][0]
    log.info(f"Created load balancing target group
'{target_group_name}'.")
    return target_group
except ClientError as err:
    log.error(
        f"Couldn't create load balancing target group
'{target_group_name}'."
    )
    error_code = err.response["Error"]["Code"]

    if error_code == "DuplicateTargetGroupName":
        log.error(
            f"Target group name {target_group_name} already exists. "
```

```
        "Check if the target group already exists."
        "Consider using a different name or deleting the existing
target group if appropriate."
    )
    elif error_code == "TooManyTargetGroups":
        log.error(
            "Too many target groups exist in the account. "
            "Consider deleting unused target groups to create space for
new ones."
        )
    log.error(f"Full error:\n\t{err}")

def delete_target_group(self, target_group_name) -> None:
    """
    Deletes the target group.
    """
    try:
        # Describe the target group to get its ARN
        response =
self.elb_client.describe_target_groups(Names=[target_group_name])
        tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

        # Delete the target group
        self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
        log.info("Deleted load balancing target group %s.",
target_group_name)

        # Use a custom waiter to wait until the target group is no longer
available
        self.wait_for_target_group_deletion(self.elb_client, tg_arn)
        log.info("Target group %s successfully deleted.", target_group_name)

    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(f"Failed to delete target group '{target_group_name}'.")
        if error_code == "TargetGroupNotFound":
            log.error(
                "Load balancer target group either already deleted or never
existed. "
                "Verify the name and check that the resource exists in the
AWS Console."
            )
        elif error_code == "ResourceInUseException":
```

```
        log.error(
            "Target group still in use by another resource. "
            "Ensure that the target group is no longer associated with
any load balancers or resources.",
        )
        log.error(f"Full error:\n\t{err}")

def wait_for_target_group_deletion(
    self, elb_client, target_group_arn, max_attempts=10, delay=30
):
    for attempt in range(max_attempts):
        try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
            print(
                f"Attempt {attempt + 1}: Target group {target_group_arn}
still exists."
            )
        except ClientError as e:
            if e.response["Error"]["Code"] == "TargetGroupNotFound":
                print(
                    f"Target group {target_group_arn} has been successfully
deleted."
                )
                return
            else:
                raise
                time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts
* delay} seconds."
        )

def create_load_balancer(
    self,
    load_balancer_name: str,
    subnet_ids: List[str],
) -> Dict[str, Any]:
    """
    Creates an Elastic Load Balancing load balancer that uses the specified
subnets
    and forwards requests to the specified target group.
    """
```

```
:param load_balancer_name: The name of the load balancer to create.
:param subnet_ids: A list of subnets to associate with the load balancer.
:return: Data about the newly created load balancer.
"""
try:
    response = self.elb_client.create_load_balancer(
        Name=load_balancer_name, Subnets=subnet_ids
    )
    load_balancer = response["LoadBalancers"][0]
    log.info(f"Created load balancer '{load_balancer_name}'.")

    waiter = self.elb_client.get_waiter("load_balancer_available")
    log.info(
        f"Waiting for load balancer '{load_balancer_name}' to be
available..."
    )
    waiter.wait(Names=[load_balancer_name])
    log.info(f"Load balancer '{load_balancer_name}' is now available!")

except ClientError as err:
    error_code = err.response["Error"]["Code"]
    log.error(
        f"Failed to create load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
    )

    if error_code == "DuplicateLoadBalancerNameException":
        log.error(
            f"A load balancer with the name '{load_balancer_name}'
already exists. "
            "Load balancer names must be unique within the AWS region. "
            "Please choose a different name and try again."
        )
    if error_code == "TooManyLoadBalancersException":
        log.error(
            "The maximum number of load balancers has been reached in
this account and region. "
            "You can delete unused load balancers or request an increase
in the service quota from AWS Support."
        )
        log.error(f"Full error:\n\t{err}")
else:
    return load_balancer
```

```

def create_listener(
    self,
    load_balancer_name: str,
    target_group: Dict[str, Any],
) -> Dict[str, Any]:
    """
    Creates a listener for the specified load balancer that forwards requests
to the
    specified target group.

    :param load_balancer_name: The name of the load balancer to create a
listener for.
    :param target_group: An existing target group that is added as a listener
to the
        load balancer.
    :return: Data about the newly created listener.
    """
    try:
        # Retrieve the load balancer ARN
        load_balancer_response = self.elb_client.describe_load_balancers(
            Names=[load_balancer_name]
        )
        load_balancer_arn = load_balancer_response["LoadBalancers"][0][
            "LoadBalancerArn"
        ]

        # Create the listener
        response = self.elb_client.create_listener(
            LoadBalancerArn=load_balancer_arn,
            Protocol=target_group["Protocol"],
            Port=target_group["Port"],
            DefaultActions=[
                {
                    "Type": "forward",
                    "TargetGroupArn": target_group["TargetGroupArn"],
                }
            ],
        )
        log.info(
            f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
        )
        return response["Listeners"][0]

```

```
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(
            f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
        )

        if error_code == "ListenerNotFoundException":
            log.error(
                f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                "Please check the load balancer name and target group
configuration."
            )
        if error_code == "InvalidConfigurationRequestException":
            log.error(
                f"The configuration provided for the listener on load
balancer '{load_balancer_name}' is invalid. "
                "Please review the provided protocol, port, and target group
settings."
            )
        log.error(f"Full error:\n\t{err}")

def delete_load_balancer(self, load_balancer_name) -> None:
    """
    Deletes a load balancer.

    :param load_balancer_name: The name of the load balancer to delete.
    """
    try:
        response = self.elb_client.describe_load_balancers(
            Names=[load_balancer_name]
        )
        lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
        self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
        log.info("Deleted load balancer %s.", load_balancer_name)
        waiter = self.elb_client.get_waiter("load_balancers_deleted")
        log.info("Waiting for load balancer to be deleted...")
        waiter.wait(Names=[load_balancer_name])
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(
```

```
        f"Couldn't delete load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
    )

    if error_code == "LoadBalancerNotFoundException":
        log.error(
            f"The load balancer '{load_balancer_name}' does not exist. "
            "Please check the name and try again."
        )
    log.error(f"Full error:\n\t{err}")

def get_endpoint(self, load_balancer_name) -> str:
    """
    Gets the HTTP endpoint of the load balancer.

    :return: The endpoint.
    """
    try:
        response = self.elb_client.describe_load_balancers(
            Names=[load_balancer_name]
        )
        return response["LoadBalancers"][0]["DNSName"]
    except ClientError as err:
        log.error(
            f"Couldn't get the endpoint for load balancer
{load_balancer_name}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "LoadBalancerNotFoundException":
            log.error(
                "Verify load balancer name and ensure it exists in the AWS
console."
            )
        log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
```



```
    """
    retries = 3
    verified = False
    while not verified and retries > 0:
        try:
            lb_response = requests.get(f"http://{endpoint}")
            log.info(
                "Got response %s from load balancer endpoint.",
                lb_response.status_code,
            )
            if lb_response.status_code == 200:
                verified = True
            else:
                retries = 0
        except requests.exceptions.ConnectionError:
            log.info(
                "Got connection error from load balancer endpoint,
retrying..."
            )
            retries -= 1
            time.sleep(10)
    return verified

def check_target_health(self, target_group_name: str) -> List[Dict[str,
Any]]:
    """
    Checks the health of the instances in the target group.

    :return: The health status of the target group.
    """
    try:
        tg_response = self.elb_client.describe_target_groups(
            Names=[target_group_name]
        )
        health_response = self.elb_client.describe_target_health(
            TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
        )
    except ClientError as err:
        log.error(f"Couldn't check health of {target_group_name} target(s).")
        error_code = err.response["Error"]["Code"]
        if error_code == "LoadBalancerNotFoundException":
            log.error(
                "Load balancer associated with the target group was not
found. "
```

```

        "Ensure the load balancer exists, is in the correct AWS
region, and "
        "that you have the necessary permissions to access it.",
    )
    elif error_code == "TargetGroupNotFoundException":
        log.error(
            "Target group was not found. "
            "Verify the target group name, check that it exists in the
correct region, "
            "and ensure it has not been deleted or created in a different
account.",
        )
        log.error(f"Full error:\n\t{err}")
    else:
        return health_response["TargetHealthDescriptions"]

```

Erstellen Sie eine Klasse, die DynamoDB zum Simulieren eines Empfehlungsservices verwendet.

```

class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
    movies,
    and songs.
    """

    def __init__(self, table_name: str, dynamodb_client: boto3.client):
        """
        Initializes the RecommendationService class with the necessary
        parameters.

        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    def create(self) -> Dict[str, Any]:
        """

```

```

    Creates a DynamoDB table to use as a recommendation service. The table
    has a
        hash key named 'MediaType' that defines the type of media recommended,
    such as
        Book or Movie, and a range key named 'ItemId' that, combined with the
    MediaType,
        forms a unique identifier for the recommended item.

    :return: Data about the newly created table.
    :raises RecommendationServiceError: If the table creation fails.
    """
    try:
        response = self.dynamodb_client.create_table(
            TableName=self.table_name,
            AttributeDefinitions=[
                {"AttributeName": "MediaType", "AttributeType": "S"},
                {"AttributeName": "ItemId", "AttributeType": "N"},
            ],
            KeySchema=[
                {"AttributeName": "MediaType", "KeyType": "HASH"},
                {"AttributeName": "ItemId", "KeyType": "RANGE"},
            ],
            ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
        )
        log.info("Creating table %s...", self.table_name)
        waiter = self.dynamodb_client.get_waiter("table_exists")
        waiter.wait(TableName=self.table_name)
        log.info("Table %s created.", self.table_name)
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceInUseException":
            log.info("Table %s exists, nothing to be done.", self.table_name)
        else:
            raise RecommendationServiceError(
                self.table_name, f"ClientError when creating table: {err}."
            )
    else:
        return response

def populate(self, data_file: str) -> None:
    """
    Populates the recommendations table from a JSON file.

    :param data_file: The path to the data file.

```

```
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
                batch = [{"PutRequest": {"Item": item}} for item in items]
                self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
                log.info(
                    "Populated table %s with items from %s.", self.table_name,
data_file
                )
            except ClientError as err:
                raise RecommendationServiceError(
                    self.table_name, f"Couldn't populate table from {data_file}:
{err}"
                )

        def destroy(self) -> None:
            """
            Deletes the recommendations table.

            :raises RecommendationServiceError: If the table deletion fails.
            """
            try:
                self.dynamodb_client.delete_table(TableName=self.table_name)
                log.info("Deleting table %s...", self.table_name)
                waiter = self.dynamodb_client.get_waiter("table_not_exists")
                waiter.wait(TableName=self.table_name)
                log.info("Table %s deleted.", self.table_name)
            except ClientError as err:
                if err.response["Error"]["Code"] == "ResourceNotFoundException":
                    log.info("Table %s does not exist, nothing to do.",
self.table_name)
                else:
                    raise RecommendationServiceError(
                        self.table_name, f"ClientError when deleting table: {err}."
                    )
```

Erstellen Sie eine Klasse, die Systems-Manager-Aktionen umschließt.

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
    to drive
    the demonstration of resilient architecture, such as failure of a dependency
    or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
        recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when
        a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
        """
```

```
"""
try:
    self.ssm_client.put_parameter(
        Name=name, Value=value, Overwrite=True, Type="String"
    )
    log.info("Setting parameter %s to '%s'.", name, value)
except ClientError as err:
    error_code = err.response["Error"]["Code"]
    log.error(f"Failed to set parameter {name}.")
    if error_code == "ParameterLimitExceeded":
        log.error(
            "The parameter limit has been exceeded. "
            "Consider deleting unused parameters or request a limit
increase."
        )
    elif error_code == "ParameterAlreadyExists":
        log.error(
            "The parameter already exists and overwrite is set to False.
"
            "Use Overwrite=True to update the parameter."
        )
    log.error(f"Full error:\n\t{err}")
```

- API-Einheiten finden Sie in der Python-Referenz (Boto3) API in AWS SDK den folgenden Themen.
 - [AttachLoadBalancerTargetGroups](#)
 - [CreateAutoScalingGroup](#)
 - [CreateInstanceProfile](#)
 - [CreateLaunchTemplate](#)
 - [CreateListener](#)
 - [CreateLoadBalancer](#)
 - [CreateTargetGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DeleteInstanceProfile](#)
 - [DeleteLaunchTemplate](#)
 - [DeleteLoadBalancer](#)

- [DeleteTargetGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAvailabilityZones](#)
- [DescribeIamInstanceProfileAssociations](#)
- [DescribeInstances](#)
- [DescribeLoadBalancers](#)
- [DescribeSubnets](#)
- [DescribeTargetGroups](#)
- [DescribeTargetHealth](#)
- [DescribeVpcs](#)
- [RebootInstances](#)
- [ReplaceIamInstanceProfileAssociation](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

Eine vollständige Liste der AWS SDK Entwicklerhandbücher und Codebeispiele finden Sie unter. [Erstellen Sie EC2 Amazon-Ressourcen mit einem AWS SDK](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK Versionen.

Überwachen Sie EC2 API Amazon-Anfragen mit Amazon CloudWatch

Sie können EC2 API Amazon-Anfragen mithilfe von Amazon CloudWatch überwachen. Amazon sammelt Rohdaten und verarbeitet sie zu lesbaren, nahezu in Echtzeit verfügbaren Metriken. Diese Kennzahlen bieten eine einfache Möglichkeit, die Nutzung und die Ergebnisse der EC2 API Amazon-Operationen im Laufe der Zeit zu verfolgen. Diese Informationen geben Ihnen einen besseren Überblick über die Leistung Ihrer Webanwendungen und ermöglichen es Ihnen, eine Vielzahl von Problemen zu identifizieren und zu diagnostizieren. Sie können auch Alarme einrichten, die auf bestimmte Schwellenwerte achten und Benachrichtigungen senden oder bestimmte Maßnahmen ergreifen, wenn diese Schwellenwerte erreicht werden.

Weitere Informationen zu CloudWatch finden Sie im [CloudWatch Amazon-Benutzerhandbuch](#).

Important

Amazon EC2 API Metrics ist eine optionale Funktion. Sie müssen den Zugriff auf diese Funktion beantragen. Weitere Informationen finden Sie unter [the section called "EC2APIAmazon-Metriken aktivieren"](#).

Inhalt

- [EC2APIAmazon-Metriken aktivieren](#)
- [EC2APIAmazon-Metriken und -Dimensionen](#)
- [Aufbewahrung metrischer Daten](#)
- [Überwachung von Anfragen, die in Ihrem Namen gestellt wurden](#)
- [Fakturierung](#)
- [Mit Amazon arbeiten CloudWatch](#)

EC2APIAmazon-Metriken aktivieren

Gehen Sie wie folgt vor, um Zugriff auf diese Funktion für Sie anzufordern AWS-Konto.

Um Zugriff auf diese Funktion anzufordern

1. [AWS Support Zentrum](#) öffnen.
2. Wählen Sie Create case (Fall erstellen) aus.
3. Wählen Sie Konto und Fakturierung aus.
4. Wählen Sie für Service die Optionen Allgemeine Informationen und Erste Schritte aus.
5. Wählen Sie als Kategorie die Option Nutzung AWS und Dienste aus.
6. Wählen Sie Next step: Additional information (Nächster Schritt: Zusätzliche Informationen).
7. Geben Sie unter Subject (Betreff) **Request access to Amazon EC2 API metrics** ein.
8. Geben Sie für Beschreibung den Text **Please grant my account access to Amazon EC2 API metrics. Related page: <https://docs.aws.amazon.com/AWSEC2/latest/APIReference/monitor.html>** ein. Geben Sie auch die Region an, auf die Sie Zugriff benötigen.
9. Klicken Sie auf Next step: Solve now or contact us (Nächster Schritt): Jetzt lösen oder Support kontaktieren).
10. Wählen Sie auf der Registerkarte Kontakt Ihre bevorzugte Kontaktsprache und Kontaktmethode aus.
11. Wählen Sie Absenden aus.

EC2APIAmazon-Metriken und -Dimensionen

Metriken

Die EC2 API Amazon-Metriken sind im AWS/EC2/API Namespace enthalten. In den folgenden Tabellen sind die Metriken aufgeführt, die für EC2 API Amazon-Anfragen verfügbar sind.

Metrik	Beschreibung
ClientErrors	Die Anzahl der fehlgeschlagenen API Anfragen, die durch Client-Fehler verursacht wurden. Diese Fehler werden normalerweise durch etwas verursacht, das der Client getan hat, z. B. durch die Angabe eines falschen oder ungültigen Parameters

Metrik	Beschreibung
	<p>in der Anforderung oder durch die Verwendung einer Aktion oder Ressource im Namen eines Benutzers, der nicht berechtigt ist, die Aktion oder Ressource zu verwenden.</p> <p>Einheit: Anzahl</p>
<code>RequestLimitExceeded</code>	<p>Wie oft die von Amazon zulässige maximale Anforderungsrate für Ihr Konto überschritten EC2 APIs wurde.</p> <p>EC2APIAmazon-Anfragen werden gedrosselt, um die Leistung des Service aufrechtzuerhalten. Wenn Ihre Anfragen gedrosselt wurden, erhalten Sie die Fehlermeldung. <code>Client.RequestLimitExceeded</code></p> <p>Einheit: Anzahl</p>
<code>ServerError</code>	<p>Die Anzahl der fehlgeschlagenen API Anfragen, die durch interne Serverfehler verursacht wurden.</p> <p>Diese Fehler werden normalerweise durch einen AWS serverseitigen Fehler, eine Ausnahme oder einen Ausfall verursacht.</p> <p>Einheit: Anzahl</p>
<code>SuccessfulCalls</code>	<p>Die Anzahl der erfolgreichen API Anfragen.</p> <p>Einheit: Anzahl</p>

Dimensionen

Die EC2 Amazon-Metriken können für alle EC2 API Aktionen gefiltert werden. Weitere Informationen zu Dimensionen finden Sie unter [CloudWatch Amazon-Konzepte](#).

Aufbewahrung metrischer Daten

EC2 API Amazon-Metriken werden in Intervallen CloudWatch von 1 Minute an gesendet. CloudWatch speichert Metrikdaten wie folgt:

- Datenpunkte mit einem Zeitraum von 60 Sekunden (1 Minute) stehen 15 Tage lang zur Verfügung
- Datenpunkte mit einem Zeitraum von 300 Sekunden (5 Minuten) sind 63 Tage lang verfügbar.
- Datenpunkte mit einem Zeitraum von 3600 Sekunden (1 Stunde) sind für 455 Tage (15 Monate) verfügbar.

Überwachung von Anfragen, die in Ihrem Namen gestellt wurden

API-Anfragen von AWS Diensten in Ihrem Namen, z. B. Anfragen von Rollen, die mit Services verknüpft sind, werden nicht auf Ihre API Drosselungslimits angerechnet und sie senden keine Messdaten CloudWatch für Ihr Konto an Amazon. Diese Anfragen können nicht überwacht werden mit CloudWatch

API-Anfragen, die in Ihrem Namen von Drittanbietern gestellt werden, werden auf Ihre API Drosselungslimits angerechnet und sie senden Messwerte CloudWatch für Ihr Konto an Amazon. Diese Anfragen können mit überwacht werden. CloudWatch

Fakturierung

Es gelten die CloudWatch Standardpreise und Gebühren. Für die Nutzung der EC2 API Amazon-Metriken fallen keine zusätzlichen Gebühren an. Weitere Informationen finden Sie unter [CloudWatch Amazon-Preise](#).

Mit Amazon arbeiten CloudWatch

Inhalt

- [CloudWatch Metriken anzeigen](#)
- [CloudWatch Alarme erstellen](#)

CloudWatch Metriken anzeigen

Gehen Sie wie folgt vor, um die EC2 API Amazon-Metriken anzuzeigen.

Voraussetzung

Sie müssen den Zugriff auf EC2 API Amazon-Metriken für Ihr Konto aktivieren. Weitere Informationen finden Sie unter [the section called “EC2APIAmazon-Metriken aktivieren”](#).

So zeigen Sie die EC2 API Amazon-Metriken mit der Konsole an

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Metriken, Alle Metriken aus.
3. Wählen Sie auf der Registerkarte Durchsuchen den API Metrik-Namespace EC2/aus.
4. Um die Metriken anzuzeigen, wählen Sie die Metrikdimension aus.

So zeigen Sie EC2 API Amazon-Metriken über die Befehlszeile an

Verwenden Sie einen der folgenden Befehle:

- [list-metrics](#) ()AWS CLI

```
aws cloudwatch list-metrics --namespace "AWS/EC2/API"
```

- [Holen- \(\) CWMetricList](#)AWS Tools for Windows PowerShell

```
Get-CWMetricList -Namespace "AWS/EC2/API"
```

CloudWatch Alarme erstellen

Sie können einen CloudWatch Alarm erstellen, der eine SNS Amazon-Nachricht sendet, wenn sich der Zustand des Alarms ändert. Ein Alarm überwacht eine Metrik über einen bestimmten, von Ihnen definierten Zeitraum. Es sendet eine Benachrichtigung zu einem SNS Thema, die auf dem Wert der Metrik im Verhältnis zu einem bestimmten Schwellenwert über mehrere Zeiträume basiert.

Sie können beispielsweise einen Alarm erstellen, der die Anzahl der DescribeInstances API Anfragen überwacht, die aufgrund von serverseitigen Fehlern fehlschlagen. Der folgende Alarm sendet eine E-Mail-Benachrichtigung, wenn die Anzahl der DescribeInstances API fehlgeschlagenen Anfragen innerhalb von 5 Minuten einen Schwellenwert von 10 serverseitigen Fehlern erreicht.

Voraussetzung

Sie müssen den Zugriff auf die EC2 API Amazon-Metriken für Ihr Konto aktivieren. Weitere Informationen finden Sie unter [the section called “EC2APIAmazon-Metriken aktivieren”](#).

Um einen Alarm für EC2 DescribeInstances API Amazon-Request-Serverfehler zu erstellen

1. Öffnen Sie die CloudWatch Konsole unter <https://console.aws.amazon.com/cloudwatch/>.
2. Wählen Sie im Navigationsbereich Alarms (Alarme) und All alarms (Alle Alarme) aus.
3. Wählen Sie Create alarm (Alarm erstellen) aus.
4. Wählen Sie Metrik auswählen und geben Sie dann Folgendes an:
 - a. Wählen Sie EC2/API.
 - b. Wählen Sie Metriken pro Aktion aus.
 - c. Aktivieren Sie das Kontrollkästchen neben dem Eintrag DescribeInstances, der sich in derselben Zeile wie der ServerErrorsMetrikname befindet.
 - d. Wählen Sie Select metric (Metrik auswählen) aus.
5. Die Seite Specify metric and conditions (Metrik und Bedingungen festlegen) wird angezeigt, die ein Diagramm und andere Informationen über die von Ihnen ausgewählte Metrik und Statistik anzeigt.
 - a. Geben Sie unter Metrik Folgendes an:
 - i. Wählen Sie für Statistic (Statistik) Sum (Summe) aus.
 - ii. Vergewissern Sie sich, dass für Zeitraum die Option 5 Minuten ausgewählt ist.
 - b. Geben Sie unter Conditions (Bedingungen) Folgendes an:
 - i. Wählen Sie für Threshold type (Schwellenwerttyp) die Option Static (Statisch) aus.
 - ii. Wählen Sie für Wann immer ServerErrors ist die Option Größer/Gleich >=.
 - iii. Für als... , geben Sie 10 ein.
 - c. Wählen Sie Weiter.
6. Die Seite Configure actions (Konfigurieren von Aktionen) wird angezeigt.
 - Geben Sie unter Benachrichtigung Folgendes an:
 - i. Wählen Sie für Trigger im Alarmzustand die Option Bei Alarm aus.

- ii. Wählen Sie unter SNS Thema auswählen die Option Bestehendes SNS Thema auswählen oder Neues Thema erstellen aus und füllen Sie die erforderlichen Felder für die Benachrichtigung aus.
 - iii. Wählen Sie Weiter.
7. Die Seite „Namen und Beschreibung hinzufügen“ wird angezeigt.
 - a. Geben Sie unter Alarmname einen Namen für Ihren Alarm ein. Der Name darf nur ASCII Zeichen enthalten.
 - b. Geben Sie unter Alarmbeschreibung eine optionale Beschreibung für Ihren Alarm ein.
 - c. Wählen Sie Weiter.
8. Die Seite „Vorschau und Erstellung“ wird angezeigt. Vergewissern Sie sich, dass die Informationen korrekt sind, und wählen Sie dann Alarm erstellen.

Weitere Informationen finden Sie unter [Verwenden von CloudWatch Amazon-Alarmen](#) im CloudWatch Amazon-Benutzerhandbuch.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.