aws

开发人员指南

Amazon Elastic Compute Cloud

Amazon Elastic Compute Cloud: 开发人员指南

Table of Contents

以编程方式访问 Amazon EC2

您可以使用 AWS Management Console 或编程接口创建和管理您的 Amazon EC2 资源。有关使用亚马逊 EC2 控制台的信息，请参阅亚马逊 EC2 用户指南。

工作方式

- 亚马逊 EC2 终端节点
- 最终一致性
- 等能性
- 请求限制

编程接口

- AWS Command Line Interface (AWS CLI)
- AWS CloudFormation
- AWS 开发工具包
- 低级 API

开始使用

- 代码示例
- Console-to-Code

监控

- AWS CloudTrail
- 监控请求

亚马逊 EC2 服务终端节点

端点是用作 AWS Web 服务入口点的 URL。Amazon EC2 支持以下终端节点类型：

- IPv4 端点

- 同时支持 IPv4 和 IPv6 的双堆栈端点
- FIPS 端点

当您发出请求时，您可以指定要使用的端点和区域。如果不指定端点，则默认使用 IPv4 端点。要使用不同的端点类型，您必须在请求中指定。有关如何执行此操作的示例，请参阅指定端点。

区域名称	区域	端点	协议
美国东部（俄亥俄州）	us-east-2	ec2.us-east-2.amazonaws.com ec2-fips.us-east-2.amazonaws.com ec2.us-east-2.api.aws	HTTP 和 HTTPS HTTPS HTTPS
美国东部（弗吉尼亚州北部）	us-east-1	ec2.us-east-1.amazonaws.com ec2-fips.us-east-1.amazonaws.com ec2.us-east-1.api.aws	HTTP 和 HTTPS HTTPS HTTPS
美国西部（加利福尼亚北部）	us-west-1	ec2.us-west-1.amazonaws.com ec2-fips.us-west-1.amazonaws.com ec2.us-west-1.api.aws	HTTP 和 HTTPS HTTPS HTTPS
美国西部（俄勒冈州）	us-west-2	ec2.us-west-2.amazonaws.com ec2-fips.us-west-2.amazonaws.com ec2.us-west-2.api.aws	HTTP 和 HTTPS HTTPS HTTPS
非洲（开普敦）	af-south-1	ec2.af-south-1.amazonaws.com ec2.af-south-1.api.aws	HTTP 和 HTTPS HTTPS

区域名称	区域	端点	协议
亚太地区（香港）	ap-east-1	ec2.ap-east-1.amazonaws.com ec2.ap-east-1.api.aws	HTTP 和 HTTPS HTTPS
亚太地区（海得拉巴）	ap-south-2	ec2.ap-south-2.amazonaws.com	HTTPS
亚太地区（雅加达）	ap-southeast-3	ec2.ap-southeast-3.amazonaws.com	HTTPS
亚太地区（墨尔本）	ap-southeast-4	ec2.ap-southeast-4.amazonaws.com	HTTPS
亚太地区（孟买）	ap-south-1	ec2.ap-south-1.amazonaws.com ec2.ap-south-1.api.aws	HTTP 和 HTTPS HTTPS
亚太地区（大阪）	ap-northeast-3	ec2.ap-northeast-3.amazonaws.com	HTTP 和 HTTPS
亚太地区（首尔）	ap-northeast-2	ec2.ap-northeast-2.amazonaws.com ec2.ap-northeast-2.api.aws	HTTP 和 HTTPS HTTPS
亚太地区（新加坡）	ap-southeast-1	ec2.ap-southeast-1.amazonaws.com ec2.ap-southeast-1.api.aws	HTTP 和 HTTPS HTTPS

区域名称	区域	端点	协议
亚太地区（悉尼）	ap-southeast-2	ec2.ap-southeast-2.amazonaws.com ec2.ap-southeast-2.api.aws	HTTP 和 HTTPS HTTPS
亚太地区（东京）	ap-northeast-1	ec2.ap-northeast-1.amazonaws.com ec2.ap-northeast-1.api.aws	HTTP 和 HTTPS HTTPS
加拿大（中部）	ca-central-1	ec2.ca-central-1.amazonaws.com ec2-fips.ca-central-1.amazonaws.com ec2.ca-central-1.api.aws	HTTP 和 HTTPS HTTPS HTTPS
加拿大西部（卡尔加里）	ca-west-1	ec2.ca-west-1.amazonaws.com ec2-fips.ca-west-1.amazonaws.com	HTTPS HTTPS
欧洲地区（法兰克福）	eu-central-1	ec2.eu-central-1.amazonaws.com ec2.eu-central-1.api.aws	HTTP 和 HTTPS HTTPS
欧洲地区（爱尔兰）	eu-west-1	ec2.eu-west-1.amazonaws.com ec2.eu-west-1.api.aws	HTTP 和 HTTPS HTTPS
欧洲地区（伦敦）	eu-west-2	ec2.eu-west-2.amazonaws.com ec2.eu-west-2.api.aws	HTTP 和 HTTPS HTTPS

区域名称	区域	端点	协议
欧洲地区（米兰）	eu-south-1	ec2.eu-south-1.amazonaws.com ec2.eu-south-1.api.aws	HTTP 和 HTTPS HTTPS
欧洲地区（巴黎）	eu-west-3	ec2.eu-west-3.amazonaws.com ec2.eu-west-3.api.aws	HTTP 和 HTTPS HTTPS
欧洲（西班牙）	eu-south-2	ec2.eu-south-2.amazonaws.com	HTTPS
欧洲地区（斯德哥尔摩）	eu-north-1	ec2.eu-north-1.amazonaws.com ec2.eu-north-1.api.aws	HTTP 和 HTTPS HTTPS
欧洲（苏黎世）	eu-central-2	ec2.eu-central-2.amazonaws.com	HTTPS
以色列（特拉维夫）	il-central-1	ec2.il-central-1.amazonaws.com	HTTPS
中东（巴林）	me-south-1	ec2.me-south-1.amazonaws.com ec2.me-south-1.api.aws	HTTP 和 HTTPS HTTPS
中东（阿联酋）	me-central-1	ec2.me-central-1.amazonaws.com	HTTPS
南美洲（圣保罗）	sa-east-1	ec2.sa-east-1.amazonaws.com ec2.sa-east-1.api.aws	HTTP 和 HTTPS HTTPS

区域名称	区域	端点	协议
AWS GovCloud（美国东部）	us-gov-east-1	ec2.us-gov-east-1.amazonaws.com ec2.us-gov-east-1.api.aws	HTTPS HTTPS
AWS GovCloud（美国西部）	us-gov-west-1	ec2.us-gov-west-1.amazonaws.com ec2.us-gov-west-1.api.aws	HTTPS HTTPS

有关区域的更多信息，请参阅 Amazon EC2 用户指南中的区域和可用区。有关 Amazon EC2 的终端节点列表，请参阅 Amazon Web Services 一般参考中的区域和终端节点。

主题

- IPv4 端点
- 双栈（IPv4 和 IPv6）端点
- 指定端点

有关 FIPS 端点的信息，请参阅《Amazon Web Services 一般参考》中的 FIPS 端点。

IPv4 端点

IPv4 端点仅支持 IPv4 流量。IPv4 端点适用于所有区域。

如果您指定通用端点 `ec2.amazonaws.com`，则我们将端点用于 `us-east-1`。要使用其他区域，请指定其关联端点。例如，如果您指定 `ec2.us-east-2.amazonaws.com` 为端点，我们会将您的请求定向到 `us-east-2` 端点。

IPv4 端点名称使用以下命名约定：

- *`service`*`.`*`region`*`.amazonaws.com`

例如，`eu-west-1` 区域的 IPv4 端点名称是 `ec2.eu-west-1.amazonaws.com`。有关 Amazon EC2 的终端节点列表，请参阅 Amazon Web Services 一般参考中的区域和终端节点。

双栈（IPv4 和 IPv6）端点

同时支持 IPv4 和 IPv6 流量的双堆栈端点。双栈终端节点仅在以下区域可用：

- us-east-1—美国东部（弗吉尼亚北部）
- us-east-2—美国东部（俄亥俄州）
- us-west-2—美国西部（俄勒冈）
- eu-west-1—欧洲（爱尔兰）
- ap-south-1—亚太地区（孟买）
- sa-east-1—南美洲（圣保罗）
- us-gov-east-1—AWS GovCloud （美国东部）
- us-gov-west-1—AWS GovCloud （美国西部）

当您向双堆栈端点发出请求时，端点 URL 解析为 IPv6 或 IPv4 地址，具体取决于您的网络和客户端使用的协议。

Amazon EC2 仅支持区域双栈终端节点，这意味着您必须在终端节点名称中指定区域。双堆栈端点名称使用以下命名约定：

- ec2.*region*.api.aws

例如，eu-west-1 区域的双堆栈端点名称是 ec2.eu-west-1.api.aws。有关 Amazon EC2 的终端节点列表，请参阅 Amazon Web Services 一般参考中的区域和终端节点。

指定端点

本节提供了一些在发出请求时如何指定端点的示例。

AWS CLI

以下示例说明如何使用为该us-east-2区域指定终端节点 AWS CLI。

- 双堆栈

```
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

以下示例说明如何使用为该us-east-2区域指定终端节点 AWS SDK for Java 2.x。

- 双堆栈

```
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

以下示例说明如何使用 AWS SDK for Java 1.x 为该eu-west-1区域指定终端节点。

- 双堆栈

```
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

以下示例说明如何使用为该us-east-1区域指定终端节点 AWS SDK for Go。

- 双堆栈

```
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")
})
```

亚马逊 EC2 API 的最终一致性

由于支持 API 的系统的分布式特性，Amazon EC2 API 遵循最终一致性模型。这意味着，您运行的影响您的 Amazon EC2 资源的 API 命令的结果可能不会立即显示在您随后运行的所有命令中。在执行紧随之前的 API 命令的 API 命令时，应记住这一点。

最终一致性可能会影响您管理资源的方式。例如，如果您运行命令来创建资源，则该资源最终将对其他命令可见。这意味着，如果您运行命令来修改或描述刚刚创建的资源，则该资源的 ID 可能未在整个系统中传播，并且您会收到一条错误消息，说明该资源不存在。

要管理最终一致性，可以执行以下操作：

- 在运行命令修改资源之前，请先确认资源的状态。使用指数退避算法运行相应的Describe命令，以确保有足够的时间让前一个命令在系统中传播。为此，请重复运行该Describe命令，从几秒钟的等待时间开始，然后逐渐增加到五分钟的等待时间。
- 即使命令返回准确的响应，也要在后续Describe命令之间增加等待时间。应用指数退避算法，从几秒钟的等待时间开始，然后逐渐增加到大约五分钟的等待时间。

最终一致性错误示例

以下是由于最终一致性而可能遇到的错误代码示例。

- `InvalidInstanceID.NotFound`

  如果您成功运行该`RunInstances`命令，然后使用响应中提供的实例 ID 立即运行另一个命令`RunInstances`，则可能会返回`InvalidInstanceID.NotFound`错误。这并不意味着该实例不存在。

  可能受到影响的一些特定命令有：

  - `DescribeInstances`：要确认实例的实际状态，请使用指数退避算法运行此命令。
  - `TerminateInstances`：要确认实例的状态，请先使用指数退避算法运行该`DescribeInstances`命令。

  > **Important**
  >
  > 如果您在运行后`InvalidInstanceID.NotFound`出现错误`TerminateInstances`，这并不意味着实例已经或将要终止。您的实例可能仍在运行。这就是为什么首先使用确认实例状态很重要的原因`DescribeInstances`。

- `InvalidGroup.NotFound`

  如果您成功运行该`CreateSecurityGroup`命令，然后使用响应中提供的安全组 ID 立即运行另一个命令`CreateSecurityGroup`，则可能会返回`InvalidGroup.NotFound`错误。要确认安全组的状态，请使用指数退避算法运行该`DescribeSecurityGroups`命令。

- `InstanceLimitExceeded`

  对于指定实例类型，您请求的实例数超过了当前实例限制所允许的数量。如果您快速启动和终止实例，则可能会意外达到此限制，因为已终止的实例在终止后的一段时间内计入您的实例限制。

确保 Amazon EC2 API 请求中的等性

当您发出变更 API 请求时，该请求通常会在操作的异步工作流完成之前返回结果。即使请求已经返回结果，操作在完成之前也可能会超时或遇到其他服务器问题。这样就很难确定请求是否成功，并且会导致进行多次重试以确保操作成功完成。但是，如果原始请求和后续重试成功，则会多次完成操作。这意味着您创建的资源可能会超出您的预期。

幂等性确保 API 请求完成不超过一次。对于幂等性请求，如果原始请求成功完成，则后续重试也会成功完成，而不会执行任何后续操作。但是，结果可能包含更新的信息，例如当前的创建状态。

内容

亚马逊 EC2 中的等性

以下 API 操作在默认情况下是等效的，不需要额外配置。默认情况下，相应的 AWS CLI 命令还支持等性。

默认情况下是等性的

- AssociateAddress
- CreateVpnConnection
- DisassociateAddress
- ReplaceNetworkAclAssociation
- TerminateInstances

以下 API 操作可选地使用客户端令牌支持等性。相应的 AWS CLI 命令还支持使用客户端令牌的等性。客户端令牌是一个唯一的、区分大小写的字符串，最多包含 64 个 ASCII 字符。要使用其中一个操作发出等效的 API 请求，请在请求中指定客户端令牌。您不应为其他 API 请求重复使用相同的客户端令牌。如果您使用相同的客户端令牌和相同的参数重试成功完成的请求，则重试成功而不执行任何进一步的操作。如果您使用相同的客户端令牌重试成功的请求，但除区域或可用区之外的一个或多个参数不同，则重试失败并显示错误`IdempotentParameterMismatch`。

使用客户端令牌的等性

- 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

等能的类型

- 区域-每个区域的请求是等效的。但是，您可以在不同的区域使用相同的请求，包括相同的客户端令牌。
- 区域 — 在一个区域的每个可用区中，请求都是等性的。例如，如果您在同一区域的两次调用中指定了相同的客户端令牌，则如果它们为AvailabilityZone参数指定了不同的值，则调用会成功。AllocateHosts

RunInstances 等能性

RunInstancesAPI 操作同时使用区域和区域等性。

使用的等性类型取决于您如何在 API 请求中指定可用区。 RunInstances 在以下情况下，该请求使用区域等性：

- 如果您使用放置数据类型中的AvailabilityZone参数明确指定可用区
- 如果您使用参数隐式指定可用区 SubnetId

如果您未明确或隐式指定可用区，则请求将使用区域等性。

区域性等性

区域等性可确保 RunInstances API 请求在一个区域的每个可用区中都是等的。这样可以确保使用相同客户端令牌的请求在一个地区的每个可用区内只能完成一次。但是，相同的客户端令牌可用于启动该地区其他可用区的实例。

例如，如果您发送了在可用区启动实例的等效请求，然后在us-east-1a可用区的请求中使用相同的客户端令牌，则我们会在每个us-east-1b可用区启动实例。如果一个或多个参数不同，则在这些可用区域中使用相同客户端令牌的后续重试要么成功返回而不执行任何进一步的操作，要么失败并IdempotentParameterMismatch出现错误。

区域等性

区域等性可确保 RunInstances API 请求在区域中具有等性。这样可以确保使用相同客户端令牌的请求在一个区域内只能完成一次。但是，具有相同客户端令牌的完全相同的请求可用于在不同区域启动实例。

例如，如果您发送一个等效请求以在该地区启动实例，然后在该us-east-1区域的请求中使用相同的客户端令牌，则我们会在每个eu-west-1区域启动实例。如果一个或多个参数不同，则在这些区域中使用相同客户端令牌的后续重试要么成功返回而不执行任何进一步的操作，要么因IdempotentParameterMismatch错误而失败。

> **Tip**
>
> 如果请求区域中的一个可用区不可用，则使用区域等性的RunInstances 请求可能会失败。要利用 AWS 基础设施提供的可用区功能，我们建议您在启动实例时使用区域等性。 RunInstances 即使请求的区域中没有其他可用区，使用区域等性并以可用区域为目标的请求也会成功。

示例

AWS CLI 命令示例

要使 AWS CLI 命令具有等性，请添加选项。--client-token

示例 1：等性

以下 allocate-hosts 命令使用了等性，因为它包含客户端令牌。

```
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
```

示例 2：运行实例区域等性

以下 run-instances 命令使用区域等性，因为它包含客户端令牌，但没有明确或隐式指定可用区。

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

示例 3：运行实例区域等性

以下 run-instances 命令使用区域等性，因为它包括客户端令牌和明确指定的可用区。

```
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 请求示例

要使 API 请求具有等性，请添加参数。ClientToken

示例 1：等性

以下 AllocateHostsAPI 请求使用了等性，因为它包含客户端令牌。

```
https://ec2.amazonaws.com/?Action=AllocateHosts
&AvailabilityZone=us-east-1b
&InstanceType=m5.large
&Quantity=1
&AutoPlacement=off
&ClientToken=550e8400-e29b-41d4-a716-446655440000
&AUTHPARAMS
```

示例 2： RunInstances 区域等性

以下 RunInstancesAPI 请求使用区域等性，因为它包含客户端令牌，但未明确或隐式指定可用区。

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

示例 3： RunInstances 区域等性

以下 RunInstancesAPI 请求使用区域等性，因为它包含客户端令牌和明确指定的可用区。

```
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
```

针对等性请求的重试建议

下表显示了幂等性 API 请求可能获得的一些常见响应，并提供了重试建议。

响应	建议	注释
200（正常）	不重试	原始请求成功完成。成功返回任何后续重试。
400 系列响应码（客户端错误）	不重试	请求存在问题，原因如下： • 它包括无效的参数或参数组合。 • 它使用您没有权限的操作或资源。 • 它使用正在改变状态的资源。 如果请求涉及正在改变状态的资源，则重试请求可能会成功。
500 系列响应码（服务器错误）	重试	该错误是由 AWS 服务器端问题引起的，通常是暂时性的。使用适当的退避策略重复发出请求。

亚马逊 EC2 API 的请求限制

Amazon EC2 会根据每个区域限制每个 AWS 账户的 EC2 API 请求。我们这样做是为了提高服务的性能，并确保所有 Amazon EC2 客户都能公平使用。限制可确保对 Amazon EC2 API 的调用不会超过允许的最大 API 请求限制。API 调用不受请求限制的约束，无论它们来自以下哪里：

- 第三方应用程序

- 命令行工具
- 亚马逊 EC2 控制台

如果您超过 API 限制限制，则会收到`RequestLimitExceeded`错误代码。

内容

如何应用节流

Amazon EC2 使用令牌存储桶算法来实现 API 限制。使用此算法，您的账户拥有一个持有特定数量的令牌的存储桶。存储桶中的代币数量代表您在任何给定秒钟的限制限制。

Amazon EC2 实施了两种类型的 API 限制：

API 限制类型

- 请求速率限制
- 资源速率限制

请求速率限制

通过请求速率限制，您发出的 API 请求数量会受到限制。您发出的每个请求都会从存储桶中删除一个令牌。例如，非 mut ating (`Describe*`) API 操作的存储桶大小为 100 个令牌，因此您在一秒钟内最多可以发出 100 个`Describe*`请求。如果您在一秒钟内超过 100 个请求，则会受到限制，而该秒内剩余的请求将失败。

存储桶会以设定的速率自动填充。如果存储桶低于其最大容量，则每秒向其添加一定数量的令牌，直到其达到最大容量。如果充值令牌到达时桶已满，则它们将被丢弃。存储桶容纳的令牌数量不能超过其最大数量。例如，非 mut ating (`Describe*`) API 操作的存储桶大小为 100 个令牌，填充速率为每秒 20 个令牌。如果您在一秒钟内发出 100 个 `Describe*` API 请求，则存储桶会立即减少为零 (0) 个令牌。然后，该存储桶每秒充满 20 个令牌，直到其最大容量达到 100 个令牌。这意味着之前空的存储桶会在 5 秒钟后达到其最大容量。

您无需等到存储桶完全装满后即可发出 API 请求。您可以在令牌被添加到存储桶时使用这些令牌。如果您立即使用充值令牌，则存储桶无法达到其最大容量。例如，控制台非变异操作的存储桶大小为 100 个代币，填充速率为每秒 10 个令牌。如果您在一秒钟内发出 100 个 API 请求来耗尽存储桶，则可以继续每秒发出 10 个 API 请求。只有当您每秒发出少于 10 个 API 请求时，存储桶才能重新填充到最大容量。

资源速率限制

某些 API 操作（例如`RunInstances`和）`TerminateInstances`，如下表所述，除了请求速率限制外，还使用资源速率限制。这些 API 操作有一个单独的资源令牌存储桶，该存储桶会根据受请求影响的资源数量而耗尽。与请求令牌存储桶一样，资源令牌存储桶的最大存储桶允许您进行爆发，而填充速率则允许您在需要的时间内保持稳定的请求速率。如果您超出某个 API 的特定存储桶限制，包括存储桶尚未充值以支持下一次 API 调用时，即使您尚未达到 API 总限额，API 的操作也会受到限制。

例如，的资源令牌存储桶大小`RunInstances`为 1000 个令牌，填充速率为每秒两个令牌。因此，您可以使用任意数量的 API 请求立即启动 1000 个实例，例如一个针对 1000 个实例的请求或针对 250 个实例的四个请求。资源令牌存储桶为空后，您最多可以每秒启动两个实例，对两个实例使用一个请求或对一个实例使用两个请求。

有关更多信息，请参阅 资源令牌存储桶大小和充值率。

节流限制

以下各节描述了请求令牌存储桶和资源令牌存储桶的大小以及填充率。

限制

- 请求代币桶大小和充值率
- 资源令牌存储桶大小和充值率

请求代币桶大小和充值率

出于限制请求速率的目的，API 操作分为以下几类：

- 非变异操作 — 检索资源相关数据的 API 操作。此类别通常包括所有`Describe*`操作，例如`DescribeRouteTables`、`DescribeImages`和`DescribeHosts`。这些 API 操作通常具有最高的 API 限制限制。
- 未经过滤和未分页的非变异操作 — 非变异 API 操作的特定子集，在不指定分页或过滤器的情况下调用这些操作时，使用较小的令牌桶中的令牌。建议您使用分页和筛选功能，以便从标准（较大的）令牌桶中扣除代币。

- 变更操作 — 创建、修改或删除资源的 API 操作。此类别通常包括所有未归类为非变更操作的 API 操作，例如`CreateVolume`、`ModifyHosts`、和。`DeleteSnapshot`与非变异 API 调用相比，这些操作的限制更低。
- 资源密集型操作 — 改变 API 操作，这些操作需要的时间最多，消耗的资源也最多。与变异操作相比，这些操作的限制还要低。它们与其他变异动作是分开进行限制的。
- 控制台非变更操作 — 从 Amazon EC2 控制台调用的非变更 API 操作。这些 API 操作与其他非变异的 API 操作是分开限制的。
- 未分类的操作 — 这些 API 操作会获得自己的代币桶大小和充值率，尽管顾名思义，它们属于其他类别之一。

下表显示了所有 AWS 地区的请求令牌存储桶大小和充值率。

API 操作类别	操作	存储桶最大容量	水桶填充率
非变异动作	• `Describe*` • `Get*`	100	20
未经过滤和未分页的非变异动作	• `DescribeInstances` • `DescribeNetworkInterfaces` • `DescribeVolumes` • `DescribeInstanceStatus` • `DescribeSnapshots`	50	10

API 操作类别	操作	存储桶最大容量	水桶填充率
	• DescribeSecurityGroups • DescribeSpotInstanceRequests		
变异动作	未归类为非变异操作的 API 操作。	200	5

API 操作类别	操作	存储桶最大容量	水桶填充率
资源密集型行动	• AuthorizeSecurityGroupIngress • CancelSpotInstanceRequests • CreateKeyPair • RequestSpotInstances • RevokeSecurityGroupIngress • CreateVpcPeeringConnection • AcceptVpcPeeringConnection • RejectVpcPeeringConnection • DeleteVpcPeeringConnection	50	5

API 操作类别	操作	存储桶最大容量	水桶填充率
控制台非变异动作	• Describe* • Get*	100	10
未分类的动作	RunInstances	5	2
	StartInstances	5	2
	CreateVpcEndpoint	4	0.3
	ModifyVpcEndpoint	4	0.3
	DeleteVpcEndpoints	4	0.3
	AcceptVpcEndpointConnections	10	1
	RejectVpcEndpointConnections	10	1
	CreateVpcEndpointServiceConfiguration	10	1
	ModifyVpcEndpointServiceConfiguration	10	1

API 操作类别	操作	存储桶最大容量	水桶填充率
	DeleteVpcEndpointServiceConfigurations	10	1
	CreateDefaultVpc	1	1
	CreateDefaultSubnet	1	1
	MoveAddressToVpc	1	1
	RestoreAddressToClassic	1	1
	DescribeMovingAddresses	1	1
	AdvertiseByoipCidr	1	0.1
	ProvisionByoipCidr	1	0.1
	DescribeByoipCidrs	1	0.5
	DeprovisionByoipCidr	1	0.1
	WithdrawByoipCidr	1	0.1

API 操作类别	操作	存储桶最大容量	水桶填充率
	DescribeReservedInstancesOfferings	10	10
	PurchaseReservedInstancesOffering	5	5
	DescribeSpotFleetRequests	50	3
	DescribeSpotFleetInstances	100	5
	DescribeSpotFleetRequestHistory	100	5
	AssociateEnclaveCertificateIamRole	10	1
	DisassociateEnclaveCertificateIamRole	10	1
	GetAssociatedEnclaveCertificateIamRoles	10	1

API 操作类别	操作	存储桶最大容量	水桶填充率
	GetConsoleScreenshot	每个账户 5 个 每个实例 2 个	每个账户 5 个 每个实例 1

资源令牌存储桶大小和充值率

下表列出了使用资源速率限制的 API 操作的资源令牌存储桶大小和填充率。

API 操作	存储桶最大容量	水桶填充率
RunInstances	1000	2
TerminateInstances	1000	20
StartInstances	1000	2
StopInstances	1000	20

监控 API 限制

您可以使用亚马逊 CloudWatch 来监控您的 Amazon EC2 API 调用，并收集和跟踪有关 API 限制的指标。您还可以创建警报，在接近达到 API 限制限制时向您发出警报。有关更多信息，请参阅 使用亚马逊监控亚马逊 EC2 API 请求 CloudWatch。

重试和指数退缩

您的应用程序可能需要重试 API 请求。例如：

- 检查资源状态是否有更新
- 枚举大量资源（例如，您的所有卷）
- 在请求因服务器错误 (5xx) 或限制错误而失败后重试请求

但是，对于客户端错误 (4xx)，您必须先修改请求以更正问题，然后再尝试请求。

资源状态更改

在开始轮询以检查状态更新之前，请留出可能完成的请求时间。例如，请等待几分钟，然后再检查您的实例是否处于活动状态。开始轮询时，请在连续请求之间使用适当的睡眠间隔来降低 API 请求的速率。为了获得最佳的效果，请使用递增或可变的睡眠间隔。

或者，您可以使用 Amazon EventBridge 通知您某些资源的状态。例如，您可以使用 EC2 实例状态更改通知事件来通知您实例的状态变化。有关更多信息，请参阅使用自动化 Amazon EC2 EventBridge。

重试

当您需要轮询或重试 API 请求时，我们建议使用指数退避算法来计算 API 调用之间的睡眠间隔。指数回退的原理是对于连续错误响应，重试等待间隔越来越长。您应该实施最长延迟间隔和最大重试次数。您也可以使用抖动（随机延迟）来防止连续的碰撞。有关更多信息，请参阅超时、重试和回退并抖动。

每个 AWS SDK 都实现了自动重试逻辑。有关更多信息，请参阅 AWS SDK 和工具参考指南中的重试行为。

请求提高限制

您可以请求提高您的 API 限制限制。 AWS 账户

请求访问此功能

1. 打开AWS Support 中心。
2. 选择创建案例。
3. 选择账户和账单。
4. 对于 “服务”，选择 “一般信息” 和 “入门”。
5. 在 “类别” 中，选择 “使用 AWS 和服务”。
6. 选择 Next step: Additional information（下一步：其他信息）。
7. 对于 Subject (主题)，请输入 **Request an increase in my Amazon EC2 API throttling limits**。
8. 对于描述，输入 **Please increase the API throttling limits for my account. Related page: https://docs.aws.amazon.com/AWSEC2/latest/APIReference/throttling.html**。还包括以下信息：
   - 您的使用案例的描述。
   - 您需要加薪的地区。
   - 出现峰值限制或使用量的一小时窗口（用于计算新的限制限制），以 UTC 表示。

9. 选择下一步：立即解决或联系我们。
10. 在 “联系我们” 选项卡上，选择您的首选联系语言和联系方式。
11. 选择提交。

使用创建 Amazon EC2 资源 AWS CLI

您可以使用命令行外壳中的 AWS Command Line Interface (AWS CLI) 来创建和管理您的 Amazon EC2 资源。 AWS CLI 提供对诸如 Amazon EC2 之类的 API 的 AWS 服务直接访问权限。

有关 Amazon EC2 命令的语法和示例，请参阅《AWS CLI 命令参考》中的 ec2。 你也可以在 github 上的 aws-cli/awscli/examples/ ec2 中找到这些示例。

了解更多有关 AWS CLI

要了解更多信息 AWS CLI，请参阅以下资源：

- AWS Command Line Interface
- AWS Command Line Interface 版本 2 的用户指南
- AWS Command Line Interface 版本 1 的用户指南

使用创建 Amazon EC2 资源 AWS CloudFormation

Amazon EC2 与 AWS CloudFormation一项服务集成，可帮助您对 AWS 资源进行建模和设置，从而减少创建和管理资源和基础设施所花费的时间。您可以创建一个描述所需 AWS 资源（例如实例和子网）的模板，并为您预 AWS CloudFormation 置和配置这些资源。

使用时 AWS CloudFormation，您可以重复使用模板来一致且重复地设置您的 Amazon EC2 资源。只需描述一次您的资源，然后在多个 AWS 账户 区域中一遍又一遍地配置相同的资源。

亚马逊 EC2 和 AWS CloudFormation 模板

要为 Amazon EC2 和相关服务预置和配置资源，您必须了解AWS CloudFormation 模板。模板是 JSON 或 YAML 格式的文本文件。这些模板描述了您将在 AWS CloudFormation 堆栈中配置的资源。如果你不熟悉 JSON 或 YAML，可以使用 D AWS CloudFormation esigner 来帮助你开始使用 AWS CloudFormation 模板。有关更多信息，请参阅什么是 AWS CloudFormation 设计器？ 在《AWS CloudFormation 用户指南》中。

适用于 Amazon EC2 的资源

计算资源

- AWS::EC2::CapacityReservation
- AWS::EC2::CapacityReservation舰队
- AWS::EC2::EC2Fleet
- AWS::EC2::EC2Fleet
- AWS::EC2::Host
- AWS::EC2::Instance
- AWS::EC2::InstanceConnect端点
- AWS::EC2::LaunchTemplate
- AWS::EC2::PlacementGroup
- AWS::EC2::SpotFleet

联网资源

- AWS::EC2::CarrierGateway

- AWS::EC2::ClientVpnAuthorizationRule
- AWS::EC2::ClientVpn端点
- AWS::EC2::ClientVpn路线
- 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::LocalGateway路线
- AWS::EC2::LocalGatewayRouteTable
- AWS::EC2::LocalGatewayRouteTableVirtualInterfaceGroupAssociation
- AWS::EC2::LocalGatewayRouteTableVpcasAsociation
- AWS::EC2::NatGateway
- AWS::EC2::NetworkInterface
- AWS::EC2::NetworkInsightsAccessScope
- AWS::EC2::NetworkInsightsAccessScopeAnalysis
- AWS::EC2::NetworkInsights分析
- AWS::EC2::NetworkInsights路径
- AWS::EC2::NetworkInterface附件

- AWS::EC2::NetworkInterface权限
- AWS::EC2::NetworkPerformanceMetricSubscription
- AWS::EC2::PrefixList
- AWS::EC2::Route
- AWS::EC2::RouteTable
- AWS::EC2::Subnet
- AWS::EC2::SubnetCidr屏蔽
- AWS::EC2::SubnetNetworkAclAssociation
- AWS::EC2::SubnetRouteTableAssociation
- AWS::EC2::TrafficMirror过滤器
- AWS::EC2::TrafficMirrorFilterRule
- AWS::EC2::TrafficMirror会话
- AWS::EC2::TrafficMirror目标
- AWS::EC2::TransitGateway
- AWS::EC2::TransitGateway附件
- AWS::EC2::TransitGatewayConnect
- AWS::EC2::TransitGatewayMulticastDomain
- AWS::EC2::TransitGatewayMulticastDomainAssociation
- AWS::EC2::TransitGatewayMulticastGroupMember
- AWS::EC2::TransitGatewayMulticastGroupSource
- AWS::EC2::TransitGatewayPeeringAttachment
- AWS::EC2::TransitGateway路线
- 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

安全资源

- AWS::EC2::KeyPair
- AWS::EC2::NetworkAcl
- AWS::EC2::NetworkAcl参赛作品
- AWS::EC2::SecurityGroup
- AWS::EC2::SecurityGroup出口
- AWS::EC2::SecurityGroup入口
- AWS::EC2::VerifiedAccess端点
- AWS::EC2::VerifiedAccess群组
- AWS::EC2::VerifiedAccess实例
- AWS::EC2::VerifiedAccessTrustProvider

存储资源

- AWS::EC2::SnapshotBlockPublicAccess
- AWS::EC2::Volume
- AWS::EC2::VolumeAttachment

了解更多关于 AWS CloudFormation

要了解更多信息 AWS CloudFormation，请参阅以下资源：

- AWS CloudFormation
- AWS CloudFormation 用户指南

使用 AWS 软件开发工具包创建 Amazon EC2 资源

AWS 为许多流行的编程语言提供软件开发套件 (SDK)。SDK 通过提供以下内容来提高开发效率：

- 您可以将其整合到应用程序中的预建组件和库
- 特定语言的工具，例如编译器和调试器
- 对服务请求进行加密签名
- 请求重试
- 错误响应处理

亚马逊 EC2 API 的代码示例

提供的代码示例 AWS 向您展示了如何使用 API 和完成特定任务。有关亚马逊 EC2 API 的示例，请参阅亚马逊 EC2 的代码示例。有关其他示例，请参阅查找 AWS 软件开发工具包的代码示例或aws-doc-sdk-examples在 github 上。

了解有关 AWS 软件开发工具包的更多信息

要了解有关 AWS 软件开发工具包的更多信息，请参阅以下资源：

- AWS SDK 和工具参考指南
- 可供构建的工具 AWS
- 什么是软件开发工具包？

适用于亚马逊 EC2 的低级 API

亚马逊 EC2 的低级 API 是亚马逊 EC2 的协议级接口。使用低级 API 时，必须正确格式化每个 HTTPS 请求，并为每个请求添加有效的数字签名。有关更多信息，请参阅《亚马逊 EC2 API 参考》中的 “向亚马逊 EC2 API 发出请求”。或者，您可以使用 S AWS DK，它代表您构造和签署请求。有关更多信息，请参阅 使用 AWS 软件开发工具包。

Amazon EC2 API 由多个服务的操作和数据类型组成。要查看每项服务的操作，请参阅 Amazon EC2 API 参考中的以下页面。

- AWS Client VPN actions
- Amazon EBS 行动
- 亚马逊 EC2 操作
- AWS Network Manager actions
- AWS Nitro Enclaves 行动
- AWS Outposts actions
- AWS PrivateLink actions
- 回收站操作
- AWS Site-to-Site VPN actions
- AWS Transit Gateway actions
- AWS Verified Access actions
- 虚拟机导入/导出操作
- 亚马逊 VPC 操作
- 亚马逊 VPC IPAM 操作
- AWS Wavelength actions

使用 Console-to-Code 为您的控制台操作生成代码

Console-to-Code 目前是面向 Amazon EC2 的预览版，可能会发生变化。仅适用于美国东部（弗吉尼亚州北部）区域。

控制台提供了一条用来创建资源和测试原型的引导式路径。如果希望大规模创建相同的资源，则您需要使用自动化代码。Console-to-Code 是 Amazon EC2 控制台的一项功能，可以帮助您开始使用自动化代码。Console-to-Code 会记录您的控制台操作，包括默认值和兼容参数。然后，它使用生成式 AI 为你想要的操作推荐你首选 infrastructure-as-code (IaC) 格式的代码。您可以将此代码用作一个起点，并对其进行自定义，以使它可用于您的特定使用案例的生产。

使用 Console-to-Code 不会产生任何额外费用。

工作方式

Console-to-Code 可以帮助您开始使用自动化代码，如下所示：

1. 您可以在控制台中执行操作，例如启动实例，或者启用详细监控。
2. Console-to-Code 会记录您的所有操作，包括控制台提供的所有默认设置和兼容参数。
3. 您可以选择要在自动化脚本中执行的操作。这些操作可以是转换操作或只读（非转换）操作，也可以同时执行这两种类型的操作。
4. 例如，“控制台到代码” 会以所需的 infrastructure-as-code (IaC) 格式生成代码。 TypeScript
5. 您可以复制代码以便在您的代码开发工具中使用，也可以下载代码以进行共享。
6. 随后，可以将代码用作自动化脚本的一个起点。您需要确认代码符合您的意图，而且参数将按预期的方式配置您的资源。您需要自定义代码，以使它们可用于您的使用案例的生产。对代码感到满意之后，您即可在自动化脚本中使用代码。

有关如何在 Amazon EC2 控制台中使用 Console-to-Code 的说明，请参阅 Console-to-Code。

限制

当使用 Console-to-Code 时，需要遵循以下说明。

支持的区域

目前只在美国东部（弗吉尼亚州北部）区域提供。

支持的代码格式

控制台到代码目前可以生成以下代码格式的 infrastructure-as-code (IaC)：

- CDK Java
- CDK Python
- CDK TypeScript
- CloudFormation json
- CloudFormation YAML

保留的操作

- 当前会话：只有在当前会话期间采取的操作才会显示在记录的操作表中。不会保留在之前的会话中执行的操作。
- 浏览器刷新：刷新浏览器标签页时，记录的操作将丢失。
- 标签页隔离：记录的操作表特定于执行操作的浏览器标签页。在一个标签页中执行的操作在另一个标签页的记录的操作表中不可见。

已记录操作表

下表列出并说明了 Console-to-Code 控制台中的已记录操作表中的各个列。

列标题	描述
控制台页面	执行的操作所在的控制台页面。
操作	API 操作。
类型	操作的类型。 • 正在变更 – 用来创建、修改或删除资源的 API 操作。

列标题	描述
	• 只读 – 用来检索资源相关数据的 API 操作（通常是所有 `Describe*` 操作）。
CLI 命令	有关所执行的操作的详细信息，包括参数和值。
Creation time	执行操作的时间。

Console-to-Code

按照以下说明，在 Amazon EC2 控制台中使用 Console-to-Code 生成代码。

要查看这些步骤的动画，请参阅 观看动画：在 Amazon EC2 控制台中使用 Console-to-Code 生成代码。

要使用 Console-to-Code 生成代码

1. 打开美国东部(弗吉尼亚州北部)区域的 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/home?region=us-east-1。

   > **Note**
   >
   > Console-to-Code 处于预览版本，目前仅在美国东部(弗吉尼亚州北部)区域提供。只有在此区域执行的操作才会被记录。

2. 使用控制台创建资源和测试原型。例如，使用控制台配置和启动实例以及启用详细监控。

   Console-to-Code 会记录您执行的每个操作。

3. 在左侧左侧导航窗格中，选择 Console-to-Code。
4. 在已记录操作表中，查看已记录的操作，并确定要包括哪些用来生成代码的操作。

   - 使用搜索字段，按特定的控制台页面或操作筛选此表。当您开始键入时，将会筛选此表。
   - 使用类型下拉列表，按所有操作、变更操作或只读操作进行筛选。

> **Note**
>
> 系统将只列出在当前会话期间执行的操作。有关更多信息，请参阅 保留的操作。

5. 选中要为其生成代码的每个操作旁边的方框。

> **Note**
>
> 一次可以选择最多 5 个操作。

6. 选择生成 {code} 代码按钮。

   按钮标签将默认为上次选择的代码格式。要选择另一种代码格式，请选择按钮旁边的箭头。

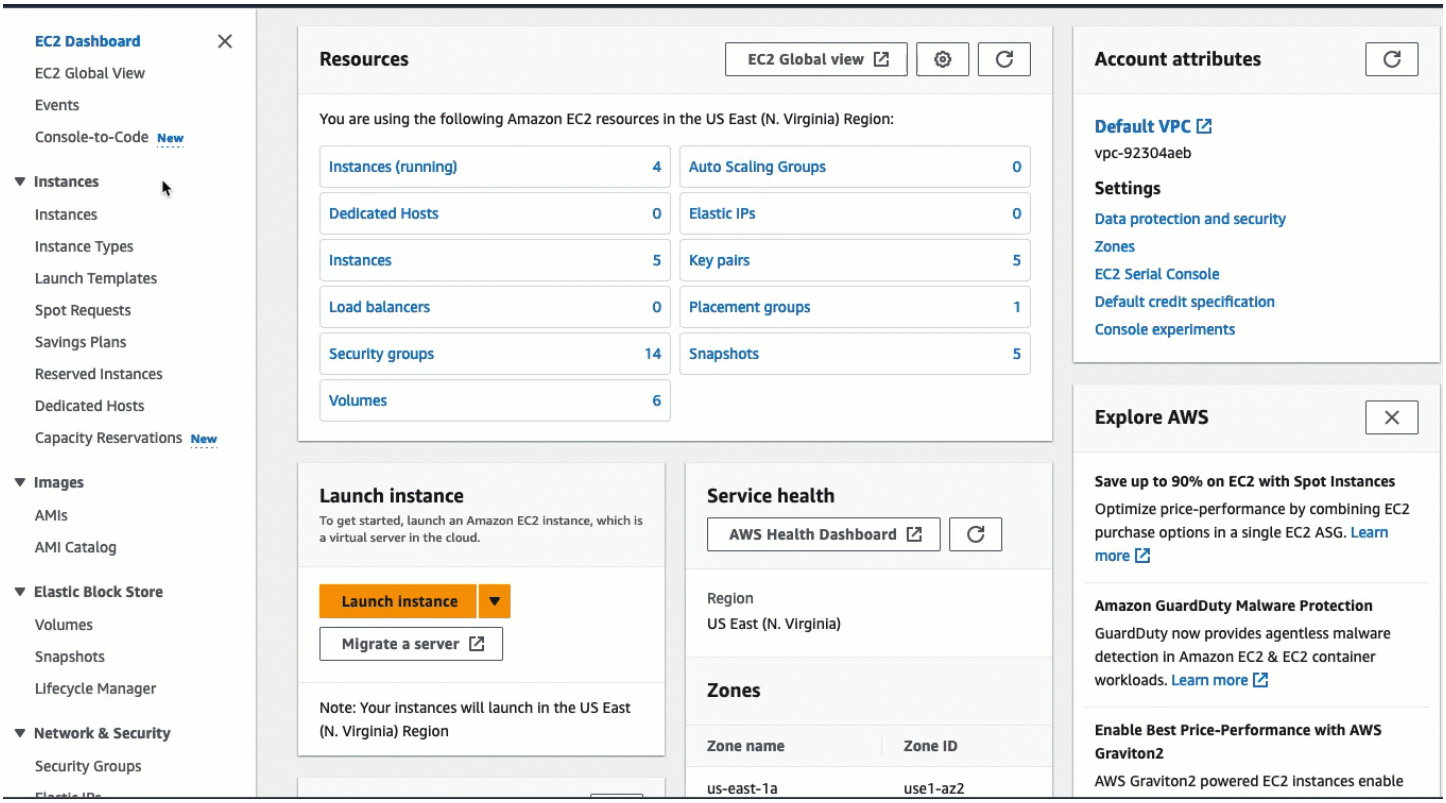
7. 在审核代码下方，选择复制以复制要在开发工具中使用的代码，或者选择下载以下载要共享的文件。

8. 使用该代码作为您的起点 infrastructure-as-code。您需要自定义代码，以使它们可用于您的特定使用案例的生产。

> **Note**
>
> 如果您发现代码尚未准备就绪，请向我们提供有关如何改进的反馈（参见以下步骤 9）。AWS Support 无法帮助您生成代码或开发自定义代码。

9. （可选）请选择拇指向上或拇指向下，以告诉我们 Console-to-Code 是否有所帮助。如果您选择拇指向下，则可以选择提供反馈，以告诉我们如何改进代码以便更好地为您提供帮助。

观看动画：在 Amazon EC2 控制台中使用 Console-to-Code 生成代码



使用 AWS 软件开发工具包的 Amazon EC2 的代码示例

以下代码示例展示了如何将 Amazon EC2 与 AWS 软件开发套件 (SDK) 配合使用。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景和跨服务示例的上下文查看操作。

场景是展示如何通过在同一服务中调用多个函数来完成特定任务任务的代码示例。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

开始使用

Hello Amazon EC2

以下代码示例展示了如何开始使用 Amazon EC2。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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>A Task object.</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>();

        var request = new DescribeSecurityGroupsRequest
        {
            MaxResults = 10,
        };


        // Retrieve information about up to 10 Amazon EC2 security groups.
        var response = await ec2Client.DescribeSecurityGroupsAsync(request);

        // Now print the security groups returned by the call to
        // DescribeSecurityGroupsAsync.
        Console.WriteLine("Security Groups:");
        response.SecurityGroups.ForEach(group =>
        {
            Console.WriteLine($"Security group: {group.GroupName} ID:
 {group.GroupId}");
        });
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考DescribeSecurityGroups中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

C MakeLists .txt cMake 文件的代码。

```
# 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})
```

hello_ec2.cpp 源文件的代码。

```
#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) {
    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) {
            auto 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;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考[DescribeSecurityGroups](中的。)

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
```

```
        try {
            DescribeSecurityGroupsRequest request =
 DescribeSecurityGroupsRequest.builder()
                .groupIds(groupId)
                .build();

            // Use a paginator.
            DescribeSecurityGroupsIterable listGroups =
 ec2.describeSecurityGroupsPaginator(request);
            listGroups.stream()
                .flatMap(r -> r.securityGroups().stream())
                .forEach(group -> System.out
                    .println(" Group id: " +group.groupId() + " group name = " +
 group.groupName()));

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[DescribeSecurityGroups](#)中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#)中查找完整实例，了解如何进行设置和运行。

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

import { client } from "./libs/client.js";

// Call DescribeSecurityGroups and display the result.
export const main = async () => {
```

```
  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);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DescribeSecurityGroups中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅适用DescribeSecurityGroups于 K otlin 的AWS SDK API 参考。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import boto3


def hello_ec2(ec2_resource):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Elastic Compute Cloud
    (Amazon EC2) resource and list the security groups in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param ec2_resource: A Boto3 EC2 ServiceResource object. This object is a
 high-level
                         resource that wraps the low-level EC2 service API.
    """
    print("Hello, Amazon EC2! Let's list up to 10 of your security groups:")
    for sg in ec2_resource.security_groups.limit(10):
        print(f"\t{sg.id}: {sg.group_name}")


if __name__ == "__main__":
```

```
    hello_ec2(boto3.resource("ec2"))
```

- 有关 API 的详细信息，请参阅适用DescribeSecurityGroups于 Python 的AWS SDK (Boto3) API 参考。

代码示例

- 使用 AWS 软件开发工具包对 Amazon EC2 执行的操作
  - AcceptVpcPeeringConnection与 AWS SDK 或 CLI 配合使用
  - AllocateAddress与 AWS SDK 或 CLI 配合使用
  - AllocateHosts与 AWS SDK 或 CLI 配合使用
  - AssignPrivateIpAddresses与 AWS SDK 或 CLI 配合使用
  - AssociateAddress与 AWS SDK 或 CLI 配合使用
  - AssociateDhcpOptions与 AWS SDK 或 CLI 配合使用
  - AssociateRouteTable与 AWS SDK 或 CLI 配合使用
  - AttachInternetGateway与 AWS SDK 或 CLI 配合使用
  - AttachNetworkInterface与 AWS SDK 或 CLI 配合使用
  - AttachVolume与 AWS SDK 或 CLI 配合使用
  - AttachVpnGateway与 AWS SDK 或 CLI 配合使用
  - AuthorizeSecurityGroupEgress与 AWS SDK 或 CLI 配合使用
  - AuthorizeSecurityGroupIngress与 AWS SDK 或 CLI 配合使用
  - CancelCapacityReservation与 AWS SDK 或 CLI 配合使用
  - CancelImportTask与 AWS SDK 或 CLI 配合使用
  - CancelSpotFleetRequests与 AWS SDK 或 CLI 配合使用
  - CancelSpotInstanceRequests与 AWS SDK 或 CLI 配合使用
  - ConfirmProductInstance与 AWS SDK 或 CLI 配合使用
  - CopyImage与 AWS SDK 或 CLI 配合使用
  - CopySnapshot与 AWS SDK 或 CLI 配合使用
  - CreateCapacityReservation与 AWS SDK 或 CLI 配合使用
  - CreateCustomerGateway与 AWS SDK 或 CLI 配合使用
  - CreateDhcpOptions与 AWS SDK 或 CLI 配合使用
  - CreateFlowLogs与 AWS SDK 或 CLI 配合使用

- CreateImage与 AWS SDK 或 CLI 配合使用
- CreateInstanceExportTask与 AWS SDK 或 CLI 配合使用
- CreateInternetGateway与 AWS SDK 或 CLI 配合使用
- CreateKeyPair与 AWS SDK 或 CLI 配合使用
- CreateLaunchTemplate与 AWS SDK 或 CLI 配合使用
- CreateNetworkAcl与 AWS SDK 或 CLI 配合使用
- CreateNetworkAclEntry与 AWS SDK 或 CLI 配合使用
- CreateNetworkInterface与 AWS SDK 或 CLI 配合使用
- CreatePlacementGroup与 AWS SDK 或 CLI 配合使用
- CreateRoute与 AWS SDK 或 CLI 配合使用
- CreateRouteTable与 AWS SDK 或 CLI 配合使用
- CreateSecurityGroup与 AWS SDK 或 CLI 配合使用
- CreateSnapshot与 AWS SDK 或 CLI 配合使用
- CreateSpotDatafeedSubscription与 AWS SDK 或 CLI 配合使用
- CreateSubnet与 AWS SDK 或 CLI 配合使用
- CreateTags与 AWS SDK 或 CLI 配合使用
- CreateVolume与 AWS SDK 或 CLI 配合使用
- CreateVpc与 AWS SDK 或 CLI 配合使用
- CreateVpcEndpoint与 AWS SDK 或 CLI 配合使用
- CreateVpnConnection与 AWS SDK 或 CLI 配合使用
- CreateVpnConnectionRoute与 AWS SDK 或 CLI 配合使用
- CreateVpnGateway与 AWS SDK 或 CLI 配合使用
- DeleteCustomerGateway与 AWS SDK 或 CLI 配合使用
- DeleteDhcpOptions与 AWS SDK 或 CLI 配合使用
- DeleteFlowLogs与 AWS SDK 或 CLI 配合使用
- DeleteInternetGateway与 AWS SDK 或 CLI 配合使用
- DeleteKeyPair与 AWS SDK 或 CLI 配合使用
- DeleteLaunchTemplate与 AWS SDK 或 CLI 配合使用
- DeleteNetworkAcl与 AWS SDK 或 CLI 配合使用
- DeleteNetworkAclEntry与 AWS SDK 或 CLI 配合使用

- DeleteNetworkInterface与 AWS SDK 或 CLI 配合使用
- DeletePlacementGroup与 AWS SDK 或 CLI 配合使用
- DeleteRoute与 AWS SDK 或 CLI 配合使用
- DeleteRouteTable与 AWS SDK 或 CLI 配合使用
- DeleteSecurityGroup与 AWS SDK 或 CLI 配合使用
- DeleteSnapshot与 AWS SDK 或 CLI 配合使用
- DeleteSpotDatafeedSubscription与 AWS SDK 或 CLI 配合使用
- DeleteSubnet与 AWS SDK 或 CLI 配合使用
- DeleteTags与 AWS SDK 或 CLI 配合使用
- DeleteVolume与 AWS SDK 或 CLI 配合使用
- DeleteVpc与 AWS SDK 或 CLI 配合使用
- DeleteVpnConnection与 AWS SDK 或 CLI 配合使用
- DeleteVpnConnectionRoute与 AWS SDK 或 CLI 配合使用
- DeleteVpnGateway与 AWS SDK 或 CLI 配合使用
- DeregisterImage与 AWS SDK 或 CLI 配合使用
- DescribeAccountAttributes与 AWS SDK 或 CLI 配合使用
- DescribeAddresses与 AWS SDK 或 CLI 配合使用
- DescribeAvailabilityZones与 AWS SDK 或 CLI 配合使用
- DescribeBundleTasks与 AWS SDK 或 CLI 配合使用
- DescribeCapacityReservations与 AWS SDK 或 CLI 配合使用
- DescribeCustomerGateways与 AWS SDK 或 CLI 配合使用
- DescribeDhcpOptions与 AWS SDK 或 CLI 配合使用
- DescribeFlowLogs与 AWS SDK 或 CLI 配合使用
- DescribeHostReservationOfferings与 AWS SDK 或 CLI 配合使用
- DescribeHosts与 AWS SDK 或 CLI 配合使用
- DescribeIamInstanceProfileAssociations与 AWS SDK 或 CLI 配合使用
- DescribeIdFormat与 AWS SDK 或 CLI 配合使用
- DescribeIdentityIdFormat与 AWS SDK 或 CLI 配合使用
- DescribeImageAttribute与 AWS SDK 或 CLI 配合使用
- DescribeImages与 AWS SDK 或 CLI 配合使用

- DescribeImportImageTasks与 AWS SDK 或 CLI 配合使用
- DescribeImportSnapshotTasks与 AWS SDK 或 CLI 配合使用
- DescribeInstanceAttribute与 AWS SDK 或 CLI 配合使用
- DescribeInstanceStatus与 AWS SDK 或 CLI 配合使用
- DescribeInstanceTypes与 AWS SDK 或 CLI 配合使用
- DescribeInstances与 AWS SDK 或 CLI 配合使用
- DescribeInternetGateways与 AWS SDK 或 CLI 配合使用
- DescribeKeyPairs与 AWS SDK 或 CLI 配合使用
- DescribeNetworkAcls与 AWS SDK 或 CLI 配合使用
- DescribeNetworkInterfaceAttribute与 AWS SDK 或 CLI 配合使用
- DescribeNetworkInterfaces与 AWS SDK 或 CLI 配合使用
- DescribePlacementGroups与 AWS SDK 或 CLI 配合使用
- DescribePrefixLists与 AWS SDK 或 CLI 配合使用
- DescribeRegions与 AWS SDK 或 CLI 配合使用
- DescribeRouteTables与 AWS SDK 或 CLI 配合使用
- DescribeScheduledInstanceAvailability与 AWS SDK 或 CLI 配合使用
- DescribeScheduledInstances与 AWS SDK 或 CLI 配合使用
- DescribeSecurityGroups与 AWS SDK 或 CLI 配合使用
- DescribeSnapshotAttribute与 AWS SDK 或 CLI 配合使用
- DescribeSnapshots与 AWS SDK 或 CLI 配合使用
- DescribeSpotDatafeedSubscription与 AWS SDK 或 CLI 配合使用
- DescribeSpotFleetInstances与 AWS SDK 或 CLI 配合使用
- DescribeSpotFleetRequestHistory与 AWS SDK 或 CLI 配合使用
- DescribeSpotFleetRequests与 AWS SDK 或 CLI 配合使用
- DescribeSpotInstanceRequests与 AWS SDK 或 CLI 配合使用
- DescribeSpotPriceHistory与 AWS SDK 或 CLI 配合使用
- DescribeSubnets与 AWS SDK 或 CLI 配合使用
- DescribeTags与 AWS SDK 或 CLI 配合使用
- DescribeVolumeAttribute与 AWS SDK 或 CLI 配合使用
- DescribeVolumeStatus与 AWS SDK 或 CLI 配合使用

- DescribeVolumes与 AWS SDK 或 CLI 配合使用
- DescribeVpcAttribute与 AWS SDK 或 CLI 配合使用
- DescribeVpcClassicLink与 AWS SDK 或 CLI 配合使用
- DescribeVpcClassicLinkDnsSupport与 AWS SDK 或 CLI 配合使用
- DescribeVpcEndpointServices与 AWS SDK 或 CLI 配合使用
- DescribeVpcEndpoints与 AWS SDK 或 CLI 配合使用
- DescribeVpcs与 AWS SDK 或 CLI 配合使用
- DescribeVpnConnections与 AWS SDK 或 CLI 配合使用
- DescribeVpnGateways与 AWS SDK 或 CLI 配合使用
- DetachInternetGateway与 AWS SDK 或 CLI 配合使用
- DetachNetworkInterface与 AWS SDK 或 CLI 配合使用
- DetachVolume与 AWS SDK 或 CLI 配合使用
- DetachVpnGateway与 AWS SDK 或 CLI 配合使用
- DisableVgwRoutePropagation与 AWS SDK 或 CLI 配合使用
- DisableVpcClassicLink与 AWS SDK 或 CLI 配合使用
- DisableVpcClassicLinkDnsSupport与 AWS SDK 或 CLI 配合使用
- DisassociateAddress与 AWS SDK 或 CLI 配合使用
- DisassociateRouteTable与 AWS SDK 或 CLI 配合使用
- EnableVgwRoutePropagation与 AWS SDK 或 CLI 配合使用
- EnableVolumeIo与 AWS SDK 或 CLI 配合使用
- EnableVpcClassicLink与 AWS SDK 或 CLI 配合使用
- EnableVpcClassicLinkDnsSupport与 AWS SDK 或 CLI 配合使用
- GetConsoleOutput与 AWS SDK 或 CLI 配合使用
- GetHostReservationPurchasePreview与 AWS SDK 或 CLI 配合使用
- GetPasswordData与 AWS SDK 或 CLI 配合使用
- ImportImage与 AWS SDK 或 CLI 配合使用
- ImportKeyPair与 AWS SDK 或 CLI 配合使用
- ImportSnapshot与 AWS SDK 或 CLI 配合使用
- ModifyCapacityReservation与 AWS SDK 或 CLI 配合使用
- ModifyHosts与 AWS SDK 或 CLI 配合使用

- ModifyIdFormat与 AWS SDK 或 CLI 配合使用
- ModifyImageAttribute与 AWS SDK 或 CLI 配合使用
- ModifyInstanceAttribute与 AWS SDK 或 CLI 配合使用
- ModifyInstanceCreditSpecification与 AWS SDK 或 CLI 配合使用
- ModifyNetworkInterfaceAttribute与 AWS SDK 或 CLI 配合使用
- ModifyReservedInstances与 AWS SDK 或 CLI 配合使用
- ModifySnapshotAttribute与 AWS SDK 或 CLI 配合使用
- ModifySpotFleetRequest与 AWS SDK 或 CLI 配合使用
- ModifySubnetAttribute与 AWS SDK 或 CLI 配合使用
- ModifyVolumeAttribute与 AWS SDK 或 CLI 配合使用
- ModifyVpcAttribute与 AWS SDK 或 CLI 配合使用
- MonitorInstances与 AWS SDK 或 CLI 配合使用
- MoveAddressToVpc与 AWS SDK 或 CLI 配合使用
- PurchaseHostReservation与 AWS SDK 或 CLI 配合使用
- PurchaseScheduledInstances与 AWS SDK 或 CLI 配合使用
- RebootInstances与 AWS SDK 或 CLI 配合使用
- RegisterImage与 AWS SDK 或 CLI 配合使用
- RejectVpcPeeringConnection与 AWS SDK 或 CLI 配合使用
- ReleaseAddress与 AWS SDK 或 CLI 配合使用
- ReleaseHosts与 AWS SDK 或 CLI 配合使用
- ReplaceIamInstanceProfileAssociation与 AWS SDK 或 CLI 配合使用
- ReplaceNetworkAclAssociation与 AWS SDK 或 CLI 配合使用
- ReplaceNetworkAclEntry与 AWS SDK 或 CLI 配合使用
- ReplaceRoute与 AWS SDK 或 CLI 配合使用
- ReplaceRouteTableAssociation与 AWS SDK 或 CLI 配合使用
- ReportInstanceStatus与 AWS SDK 或 CLI 配合使用
- RequestSpotFleet与 AWS SDK 或 CLI 配合使用
- RequestSpotInstances与 AWS SDK 或 CLI 配合使用
- ResetImageAttribute与 AWS SDK 或 CLI 配合使用
- ResetInstanceAttribute与 AWS SDK 或 CLI 配合使用

- ResetNetworkInterfaceAttribute与 AWS SDK 或 CLI 配合使用
- ResetSnapshotAttribute与 AWS SDK 或 CLI 配合使用
- RevokeSecurityGroupEgress与 AWS SDK 或 CLI 配合使用
- RevokeSecurityGroupIngress与 AWS SDK 或 CLI 配合使用
- RunInstances与 AWS SDK 或 CLI 配合使用
- RunScheduledInstances与 AWS SDK 或 CLI 配合使用
- StartInstances与 AWS SDK 或 CLI 配合使用
- StopInstances与 AWS SDK 或 CLI 配合使用
- TerminateInstances与 AWS SDK 或 CLI 配合使用
- UnassignPrivateIpAddresses与 AWS SDK 或 CLI 配合使用
- UnmonitorInstances与 AWS SDK 或 CLI 配合使用

- 使用 AWS 软件开发工具包的 Amazon EC2 场景
  - 使用 AWS SDK 构建和管理弹性服务
  - 使用 AWS 软件开发工具包开始使用 Amazon EC2 实例

使用 AWS 软件开发工具包对 Amazon EC2 执行的操作

以下代码示例演示了如何使用 AWS 软件开发工具包执行单个 Amazon EC2 操作。这些代码节选调用了 Amazon EC2 API，是必须在上下文中运行的较大型程序的代码节选。每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关设置和运行代码的说明。

以下示例仅包括最常用的操作。有关完整列表，请参阅 Amazon Elastic Compute Cloud（Amazon EC2）API 参考。

示例

- AcceptVpcPeeringConnection与 AWS SDK 或 CLI 配合使用
- AllocateAddress与 AWS SDK 或 CLI 配合使用
- AllocateHosts与 AWS SDK 或 CLI 配合使用
- AssignPrivateIpAddresses与 AWS SDK 或 CLI 配合使用
- AssociateAddress与 AWS SDK 或 CLI 配合使用
- AssociateDhcpOptions与 AWS SDK 或 CLI 配合使用
- AssociateRouteTable与 AWS SDK 或 CLI 配合使用
- AttachInternetGateway与 AWS SDK 或 CLI 配合使用

- AttachNetworkInterface与 AWS SDK 或 CLI 配合使用
- AttachVolume与 AWS SDK 或 CLI 配合使用
- AttachVpnGateway与 AWS SDK 或 CLI 配合使用
- AuthorizeSecurityGroupEgress与 AWS SDK 或 CLI 配合使用
- AuthorizeSecurityGroupIngress与 AWS SDK 或 CLI 配合使用
- CancelCapacityReservation与 AWS SDK 或 CLI 配合使用
- CancelImportTask与 AWS SDK 或 CLI 配合使用
- CancelSpotFleetRequests与 AWS SDK 或 CLI 配合使用
- CancelSpotInstanceRequests与 AWS SDK 或 CLI 配合使用
- ConfirmProductInstance与 AWS SDK 或 CLI 配合使用
- CopyImage与 AWS SDK 或 CLI 配合使用
- CopySnapshot与 AWS SDK 或 CLI 配合使用
- CreateCapacityReservation与 AWS SDK 或 CLI 配合使用
- CreateCustomerGateway与 AWS SDK 或 CLI 配合使用
- CreateDhcpOptions与 AWS SDK 或 CLI 配合使用
- CreateFlowLogs与 AWS SDK 或 CLI 配合使用
- CreateImage与 AWS SDK 或 CLI 配合使用
- CreateInstanceExportTask与 AWS SDK 或 CLI 配合使用
- CreateInternetGateway与 AWS SDK 或 CLI 配合使用
- CreateKeyPair与 AWS SDK 或 CLI 配合使用
- CreateLaunchTemplate与 AWS SDK 或 CLI 配合使用
- CreateNetworkAcl与 AWS SDK 或 CLI 配合使用
- CreateNetworkAclEntry与 AWS SDK 或 CLI 配合使用
- CreateNetworkInterface与 AWS SDK 或 CLI 配合使用
- CreatePlacementGroup与 AWS SDK 或 CLI 配合使用
- CreateRoute与 AWS SDK 或 CLI 配合使用
- CreateRouteTable与 AWS SDK 或 CLI 配合使用
- CreateSecurityGroup与 AWS SDK 或 CLI 配合使用
- CreateSnapshot与 AWS SDK 或 CLI 配合使用
- CreateSpotDatafeedSubscription与 AWS SDK 或 CLI 配合使用

- CreateSubnet与 AWS SDK 或 CLI 配合使用
- CreateTags与 AWS SDK 或 CLI 配合使用
- CreateVolume与 AWS SDK 或 CLI 配合使用
- CreateVpc与 AWS SDK 或 CLI 配合使用
- CreateVpcEndpoint与 AWS SDK 或 CLI 配合使用
- CreateVpnConnection与 AWS SDK 或 CLI 配合使用
- CreateVpnConnectionRoute与 AWS SDK 或 CLI 配合使用
- CreateVpnGateway与 AWS SDK 或 CLI 配合使用
- DeleteCustomerGateway与 AWS SDK 或 CLI 配合使用
- DeleteDhcpOptions与 AWS SDK 或 CLI 配合使用
- DeleteFlowLogs与 AWS SDK 或 CLI 配合使用
- DeleteInternetGateway与 AWS SDK 或 CLI 配合使用
- DeleteKeyPair与 AWS SDK 或 CLI 配合使用
- DeleteLaunchTemplate与 AWS SDK 或 CLI 配合使用
- DeleteNetworkAcl与 AWS SDK 或 CLI 配合使用
- DeleteNetworkAclEntry与 AWS SDK 或 CLI 配合使用
- DeleteNetworkInterface与 AWS SDK 或 CLI 配合使用
- DeletePlacementGroup与 AWS SDK 或 CLI 配合使用
- DeleteRoute与 AWS SDK 或 CLI 配合使用
- DeleteRouteTable与 AWS SDK 或 CLI 配合使用
- DeleteSecurityGroup与 AWS SDK 或 CLI 配合使用
- DeleteSnapshot与 AWS SDK 或 CLI 配合使用
- DeleteSpotDatafeedSubscription与 AWS SDK 或 CLI 配合使用
- DeleteSubnet与 AWS SDK 或 CLI 配合使用
- DeleteTags与 AWS SDK 或 CLI 配合使用
- DeleteVolume与 AWS SDK 或 CLI 配合使用
- DeleteVpc与 AWS SDK 或 CLI 配合使用
- DeleteVpnConnection与 AWS SDK 或 CLI 配合使用
- DeleteVpnConnectionRoute与 AWS SDK 或 CLI 配合使用
- DeleteVpnGateway与 AWS SDK 或 CLI 配合使用

- DeregisterImage与 AWS SDK 或 CLI 配合使用
- DescribeAccountAttributes与 AWS SDK 或 CLI 配合使用
- DescribeAddresses与 AWS SDK 或 CLI 配合使用
- DescribeAvailabilityZones与 AWS SDK 或 CLI 配合使用
- DescribeBundleTasks与 AWS SDK 或 CLI 配合使用
- DescribeCapacityReservations与 AWS SDK 或 CLI 配合使用
- DescribeCustomerGateways与 AWS SDK 或 CLI 配合使用
- DescribeDhcpOptions与 AWS SDK 或 CLI 配合使用
- DescribeFlowLogs与 AWS SDK 或 CLI 配合使用
- DescribeHostReservationOfferings与 AWS SDK 或 CLI 配合使用
- DescribeHosts与 AWS SDK 或 CLI 配合使用
- DescribeIamInstanceProfileAssociations与 AWS SDK 或 CLI 配合使用
- DescribeIdFormat与 AWS SDK 或 CLI 配合使用
- DescribeIdentityIdFormat与 AWS SDK 或 CLI 配合使用
- DescribeImageAttribute与 AWS SDK 或 CLI 配合使用
- DescribeImages与 AWS SDK 或 CLI 配合使用
- DescribeImportImageTasks与 AWS SDK 或 CLI 配合使用
- DescribeImportSnapshotTasks与 AWS SDK 或 CLI 配合使用
- DescribeInstanceAttribute与 AWS SDK 或 CLI 配合使用
- DescribeInstanceStatus与 AWS SDK 或 CLI 配合使用
- DescribeInstanceTypes与 AWS SDK 或 CLI 配合使用
- DescribeInstances与 AWS SDK 或 CLI 配合使用
- DescribeInternetGateways与 AWS SDK 或 CLI 配合使用
- DescribeKeyPairs与 AWS SDK 或 CLI 配合使用
- DescribeNetworkAcls与 AWS SDK 或 CLI 配合使用
- DescribeNetworkInterfaceAttribute与 AWS SDK 或 CLI 配合使用
- DescribeNetworkInterfaces与 AWS SDK 或 CLI 配合使用
- DescribePlacementGroups与 AWS SDK 或 CLI 配合使用
- DescribePrefixLists与 AWS SDK 或 CLI 配合使用
- DescribeRegions与 AWS SDK 或 CLI 配合使用

- DescribeRouteTables与 AWS SDK 或 CLI 配合使用
- DescribeScheduledInstanceAvailability与 AWS SDK 或 CLI 配合使用
- DescribeScheduledInstances与 AWS SDK 或 CLI 配合使用
- DescribeSecurityGroups与 AWS SDK 或 CLI 配合使用
- DescribeSnapshotAttribute与 AWS SDK 或 CLI 配合使用
- DescribeSnapshots与 AWS SDK 或 CLI 配合使用
- DescribeSpotDatafeedSubscription与 AWS SDK 或 CLI 配合使用
- DescribeSpotFleetInstances与 AWS SDK 或 CLI 配合使用
- DescribeSpotFleetRequestHistory与 AWS SDK 或 CLI 配合使用
- DescribeSpotFleetRequests与 AWS SDK 或 CLI 配合使用
- DescribeSpotInstanceRequests与 AWS SDK 或 CLI 配合使用
- DescribeSpotPriceHistory与 AWS SDK 或 CLI 配合使用
- DescribeSubnets与 AWS SDK 或 CLI 配合使用
- DescribeTags与 AWS SDK 或 CLI 配合使用
- DescribeVolumeAttribute与 AWS SDK 或 CLI 配合使用
- DescribeVolumeStatus与 AWS SDK 或 CLI 配合使用
- DescribeVolumes与 AWS SDK 或 CLI 配合使用
- DescribeVpcAttribute与 AWS SDK 或 CLI 配合使用
- DescribeVpcClassicLink与 AWS SDK 或 CLI 配合使用
- DescribeVpcClassicLinkDnsSupport与 AWS SDK 或 CLI 配合使用
- DescribeVpcEndpointServices与 AWS SDK 或 CLI 配合使用
- DescribeVpcEndpoints与 AWS SDK 或 CLI 配合使用
- DescribeVpcs与 AWS SDK 或 CLI 配合使用
- DescribeVpnConnections与 AWS SDK 或 CLI 配合使用
- DescribeVpnGateways与 AWS SDK 或 CLI 配合使用
- DetachInternetGateway与 AWS SDK 或 CLI 配合使用
- DetachNetworkInterface与 AWS SDK 或 CLI 配合使用
- DetachVolume与 AWS SDK 或 CLI 配合使用
- DetachVpnGateway与 AWS SDK 或 CLI 配合使用
- DisableVgwRoutePropagation与 AWS SDK 或 CLI 配合使用

- DisableVpcClassicLink与 AWS SDK 或 CLI 配合使用
- DisableVpcClassicLinkDnsSupport与 AWS SDK 或 CLI 配合使用
- DisassociateAddress与 AWS SDK 或 CLI 配合使用
- DisassociateRouteTable与 AWS SDK 或 CLI 配合使用
- EnableVgwRoutePropagation与 AWS SDK 或 CLI 配合使用
- EnableVolumeIo与 AWS SDK 或 CLI 配合使用
- EnableVpcClassicLink与 AWS SDK 或 CLI 配合使用
- EnableVpcClassicLinkDnsSupport与 AWS SDK 或 CLI 配合使用
- GetConsoleOutput与 AWS SDK 或 CLI 配合使用
- GetHostReservationPurchasePreview与 AWS SDK 或 CLI 配合使用
- GetPasswordData与 AWS SDK 或 CLI 配合使用
- ImportImage与 AWS SDK 或 CLI 配合使用
- ImportKeyPair与 AWS SDK 或 CLI 配合使用
- ImportSnapshot与 AWS SDK 或 CLI 配合使用
- ModifyCapacityReservation与 AWS SDK 或 CLI 配合使用
- ModifyHosts与 AWS SDK 或 CLI 配合使用
- ModifyIdFormat与 AWS SDK 或 CLI 配合使用
- ModifyImageAttribute与 AWS SDK 或 CLI 配合使用
- ModifyInstanceAttribute与 AWS SDK 或 CLI 配合使用
- ModifyInstanceCreditSpecification与 AWS SDK 或 CLI 配合使用
- ModifyNetworkInterfaceAttribute与 AWS SDK 或 CLI 配合使用
- ModifyReservedInstances与 AWS SDK 或 CLI 配合使用
- ModifySnapshotAttribute与 AWS SDK 或 CLI 配合使用
- ModifySpotFleetRequest与 AWS SDK 或 CLI 配合使用
- ModifySubnetAttribute与 AWS SDK 或 CLI 配合使用
- ModifyVolumeAttribute与 AWS SDK 或 CLI 配合使用
- ModifyVpcAttribute与 AWS SDK 或 CLI 配合使用
- MonitorInstances与 AWS SDK 或 CLI 配合使用
- MoveAddressToVpc与 AWS SDK 或 CLI 配合使用
- PurchaseHostReservation与 AWS SDK 或 CLI 配合使用

- PurchaseScheduledInstances与 AWS SDK 或 CLI 配合使用
- RebootInstances与 AWS SDK 或 CLI 配合使用
- RegisterImage与 AWS SDK 或 CLI 配合使用
- RejectVpcPeeringConnection与 AWS SDK 或 CLI 配合使用
- ReleaseAddress与 AWS SDK 或 CLI 配合使用
- ReleaseHosts与 AWS SDK 或 CLI 配合使用
- ReplaceIamInstanceProfileAssociation与 AWS SDK 或 CLI 配合使用
- ReplaceNetworkAclAssociation与 AWS SDK 或 CLI 配合使用
- ReplaceNetworkAclEntry与 AWS SDK 或 CLI 配合使用
- ReplaceRoute与 AWS SDK 或 CLI 配合使用
- ReplaceRouteTableAssociation与 AWS SDK 或 CLI 配合使用
- ReportInstanceStatus与 AWS SDK 或 CLI 配合使用
- RequestSpotFleet与 AWS SDK 或 CLI 配合使用
- RequestSpotInstances与 AWS SDK 或 CLI 配合使用
- ResetImageAttribute与 AWS SDK 或 CLI 配合使用
- ResetInstanceAttribute与 AWS SDK 或 CLI 配合使用
- ResetNetworkInterfaceAttribute与 AWS SDK 或 CLI 配合使用
- ResetSnapshotAttribute与 AWS SDK 或 CLI 配合使用
- RevokeSecurityGroupEgress与 AWS SDK 或 CLI 配合使用
- RevokeSecurityGroupIngress与 AWS SDK 或 CLI 配合使用
- RunInstances与 AWS SDK 或 CLI 配合使用
- RunScheduledInstances与 AWS SDK 或 CLI 配合使用
- StartInstances与 AWS SDK 或 CLI 配合使用
- StopInstances与 AWS SDK 或 CLI 配合使用
- TerminateInstances与 AWS SDK 或 CLI 配合使用
- UnassignPrivateIpAddresses与 AWS SDK 或 CLI 配合使用
- UnmonitorInstances与 AWS SDK 或 CLI 配合使用

`AcceptVpcPeeringConnection`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AcceptVpcPeeringConnection`。

CLI

AWS CLI

接受 VPC 对等连接

此示例接受指定的 VPC 对等连接请求。

命令:

```
aws ec2 accept-vpc-peering-connection --vpc-peering-connection-id pcx-1a2b3c4d
```

输出：

```
{
  "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"
    }
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考AcceptVpcPeeringConnection中的。

PowerShell

用于 PowerShell

示例 1：此示例批准了请求的 VpcPeeringConnectionId pcx-1dfad234b56ff78be

```
Approve-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-1dfad234b56ff78be
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考AcceptVpcPeeringConnection中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AllocateAddress`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AllocateAddress`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用实例

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
/// <summary>
/// Allocate an Elastic IP address.
/// </summary>
```

```
    /// <returns>The allocation Id of the allocated address.</returns>
    public async Task<string> AllocateAddress()
    {
        var request = new AllocateAddressRequest();

        var response = await _amazonEC2.AllocateAddressAsync(request);
        return response.AllocationId;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考AllocateAddress中的。

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
###############################################################################
# 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
}
```

本示例中使用的实用程序函数。

```
###############################################################################
# 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
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考AllocateAddress中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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;
    }

    allocationId = outcome.GetResult().GetAllocationId();
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考AllocateAddress中的。

CLI

AWS CLI

示例 1：从 Amazon 的地址池中分配弹性 IP 地址

以下 `allocate-address` 示例分配弹性 IP 地址。Amazon EC2 从 Amazon 的地址池中选择地址。

```
aws ec2 allocate-address
```

输出：

```
{
    "PublicIp": "70.224.234.241",
    "AllocationId": "eipalloc-01435ba59eEXAMPLE",
    "PublicIpv4Pool": "amazon",
    "NetworkBorderGroup": "us-west-2",
    "Domain": "vpc"
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的弹性 IP 地址。

示例 2：分配弹性 IP 地址并将其与网络边界组关联

以下 `allocate-address` 示例分配弹性 IP 地址并将其与指定的网络边界组关联。

```
aws ec2 allocate-address \
    --network-border-group us-west-2-lax-1
```

输出：

```
{
    "PublicIp": "70.224.234.241",
    "AllocationId": "eipalloc-e03dd489ceEXAMPLE",
    "PublicIpv4Pool": "amazon",
    "NetworkBorderGroup": "us-west-2-lax-1",
    "Domain": "vpc"
```

```
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的弹性 IP 地址。

示例 3：从自己拥有的地址池中分配弹性 IP 地址

以下 `allocate-address` 示例从您引入 Amazon Web Services 账户的地址池中，分配弹性 IP 地址。Amazon EC2 从地址池中选择地址。

```
aws ec2 allocate-address \
    --public-ipv4-pool ipv4pool-ec2-1234567890abcdef0
```

输出：

```
{
    "AllocationId": "eipalloc-02463d08ceEXAMPLE",
    "NetworkBorderGroup": "us-west-2",
    "CustomerOwnedIp": "18.218.95.81",
    "CustomerOwnedIpv4Pool": "ipv4pool-ec2-1234567890abcdef0",
    "Domain": "vpc"
    "NetworkBorderGroup": "us-west-2",
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的弹性 IP 地址。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考AllocateAddress中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    public static String allocateAddress(Ec2Client ec2) {
        try {
            AllocateAddressRequest allocateRequest =
AllocateAddressRequest.builder()
```

```
                    .domain(DomainType.VPC)
                    .build();

            AllocateAddressResponse allocateResponse =
 ec2.allocateAddress(allocateRequest);
            return allocateResponse.allocationId();

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考AllocateAddress中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { AllocateAddressCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

export const main = async () => {
  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 (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考AllocateAddress中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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
    }
}
```

- 有关 API 的详细信息，请参阅适用AllocateAddress于 K otlin 的AWS SDK API 参考。

PowerShell

用于 PowerShell

示例 1：此示例分配弹性 IP 地址以用于 VPC 中的实例。

```
New-EC2Address -Domain Vpc
```

输出：

```
AllocationId         Domain      PublicIp
------------         ------      --------
eipalloc-12345678    vpc         198.51.100.2
```

示例 2：此示例分配了一个弹性 IP 地址以用于 EC2-Classic 中的实例。

```
New-EC2Address
```

输出：

```
AllocationId         Domain      PublicIp
------------         ------      --------
                     standard    203.0.113.17
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[AllocateAddress](#)中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions."""
```

```
   def __init__(self, ec2_resource, elastic_ip=None):
       """
       :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
resource
                            is used to create additional high-level objects
                            that wrap low-level Amazon EC2 service actions.
       :param elastic_ip: A Boto3 VpcAddress object. This is a high-level object
that
                          wraps Elastic IP actions.
       """
       self.ec2_resource = ec2_resource
       self.elastic_ip = elastic_ip

   @classmethod
   def from_resource(cls):
       ec2_resource = boto3.resource("ec2")
       return cls(ec2_resource)


   def allocate(self):
       """
       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 newly created Elastic IP object. By default, the address is
not
                associated with any instance.
       """
       try:
           response =
self.ec2_resource.meta.client.allocate_address(Domain="vpc")
           self.elastic_ip =
self.ec2_resource.VpcAddress(response["AllocationId"])
       except ClientError as err:
           logger.error(
               "Couldn't allocate Elastic IP. Here's why: %s: %s",
               err.response["Error"]["Code"],
               err.response["Error"]["Message"],
           )
           raise
       else:
```

```
            return self.elastic_ip
```

- 有关 API 的详细信息，请参阅适用AllocateAddress于 Python 的AWS SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
# 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")
  return response.allocation_id
rescue StandardError => e
  puts "Error allocating Elastic IP address: #{e.message}"
  return "Error"
end
```

- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考AllocateAddress中的。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用AllocateAddress于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AllocateHosts`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AllocateHosts`。

CLI

AWS CLI

示例 1：分配专用主机

以下`allocate-hosts`示例在`eu-west-1a`可用区中分配了一台专用主机，您可以在该主机上启动`m5.large`实例。默认情况下，专用主机仅接受目标实例启动，不支持主机恢复。

```
aws ec2 allocate-hosts \
```

```
    --instance-type m5.large \
    --availability-zone eu-west-1a \
    --quantity 1
```

输出：

```
{
    "HostIds": [
        "h-07879acf49EXAMPLE"
    ]
}
```

示例 2：分配启用自动放置和主机恢复的专用主机

以下allocate-hosts示例在eu-west-1a可用区中分配一台启用了自动放置和主机恢复的专用主机。

```
aws ec2 allocate-hosts \
    --instance-type m5.large \
    --availability-zone eu-west-1a \
    --auto-placement on \
    --host-recovery on \
    --quantity 1
```

输出：

```
{
     "HostIds": [
         "h-07879acf49EXAMPLE"
     ]
}
```

示例 3：分配带有标签的专用主机

以下allocate-hosts示例分配了一台专用主机，并应用了密钥名为purpose、值为的production标签。

```
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}'
```

输出：

```
{
    "HostIds": [
        "h-07879acf49EXAMPLE"
    ]
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的分配专用主机。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考AllocateHosts中的。

PowerShell

用于 PowerShell

示例 1：此示例为您的账户分配给定实例类型和可用区的专用主机

```
New-EC2Host -AutoPlacement on -AvailabilityZone eu-west-1b -InstanceType
 m4.xlarge -Quantity 1
```

输出：

```
h-01e23f4cd567890f3
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考AllocateHosts中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AssignPrivateIpAddresses`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AssignPrivateIpAddresses`。

CLI

AWS CLI

为特定的辅助私有 IP 地址分配网络接口

此示例将指定的辅助私有 IP 地址分配给指定的网络接口。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 assign-private-ip-addresses --network-interface-id eni-e5aa89a3 --
private-ip-addresses 10.0.0.82
```

将 Amazon EC2 选择的辅助私有 IP 地址分配给网络接口

此示例为指定的网络接口分配两个辅助私有 IP 地址。Amazon EC2 会自动从与网络接口关联的子网的 CIDR 块范围内的可用 IP 地址中分配这些 IP 地址。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 assign-private-ip-addresses --network-interface-id eni-e5aa89a3 --
secondary-private-ip-address-count 2
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考AssignPrivateIpAddresses中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的辅助私有 IP 地址分配给指定的网络接口。

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

示例 2：此示例创建了两个辅助私有 IP 地址并将它们分配给指定的网络接口。

```
Register-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -
SecondaryPrivateIpAddressCount 2
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考AssignPrivateIpAddresses中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AssociateAddress`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AssociateAddress`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用实例

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    /// <summary>
    /// Associate an Elastic IP address to an EC2 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)
    {
        var request = new AssociateAddressRequest
        {
            AllocationId = allocationId,
            InstanceId = instanceId
        };

        var response = await _amazonEC2.AssociateAddressAsync(request);
```

```
        return response.AssociationId;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考[AssociateAddress](#)中的。

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
#############################################################################
# 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
}
```

本示例中使用的实用程序函数。

```
################################################################################
# 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
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考AssociateAddress中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::AssociateAddressRequest associate_request;
    associate_request.SetInstanceId(instanceId);
    associate_request.SetAllocationId(allocationId);

    const Aws::EC2::Model::AssociateAddressOutcome associate_outcome =
            ec2Client.AssociateAddress(associate_request);
    if (!associate_outcome.IsSuccess()) {
        std::cerr << "Failed to associate Elastic IP address " << allocationId
                  << " with instance " << instanceId << ":" <<
                  associate_outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    std::cout << "Successfully associated Elastic IP address " << allocationId
              << " with instance " << instanceId << std::endl;
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考AssociateAddress中的。

CLI

AWS CLI

在 EC2-Classic 中关联弹性 IP 地址

本示例将弹性 IP 地址与 EC2-Classic 中的实例相关联。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 associate-address --instance-id i-07ffe74c7330ebf53 --public-ip
 198.51.100.0
```

关联 EC2-VPC 中的弹性 IP 地址

本示例将弹性 IP 地址与 VPC 中的实例相关联。

命令:

```
aws ec2 associate-address --instance-id i-0b263919b6498b123 --allocation-id
 eipalloc-64d5890a
```

输出：

```
{
    "AssociationId": "eipassoc-2bebb745"
}
```

本示例将弹性 IP 地址与网络接口相关联。

命令:

```
aws ec2 associate-address --allocation-id eipalloc-64d5890a --network-interface-
id eni-1a2b3c4d
```

本示例将弹性 IP 与和网络接口关联的私有 IP 地址相关联。

命令:

```
aws ec2 associate-address --allocation-id eipalloc-64d5890a --network-interface-
id eni-1a2b3c4d --private-ip-address 10.0.0.85
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考AssociateAddress中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
   public static String associateAddress(Ec2Client ec2, String instanceId,
String allocationId) {
       try {
           AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
                   .instanceId(instanceId)
                   .allocationId(allocationId)
                   .build();

           AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
           return associateResponse.associationId();

       } catch (Ec2Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考AssociateAddress中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { AssociateAddressCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

export const main = async () => {
  // You need to allocate an Elastic IP address before associating it with an
 instance.
  // You can do that with the AllocateAddressCommand.
  const allocationId = "ALLOCATION_ID";
  // You need to create an EC2 instance before an IP address can be associated
 with it.
  // You can do that with the RunInstancesCommand.
  const instanceId = "INSTANCE_ID";
  const command = new AssociateAddressCommand({
    AllocationId: allocationId,
    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 (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考AssociateAddress中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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
    }
}
```

- 有关 API 的详细信息，请参阅适用AssociateAddress于 K otlin 的AWS SDK API 参考。

PowerShell

用于 PowerShell

示例 1：此示例将指定的弹性 IP 地址与 VPC 中的指定实例相关联。

```
C:\> Register-EC2Address -InstanceId i-12345678 -AllocationId eipalloc-12345678
```

输出：

```
eipassoc-12345678
```

示例 2：此示例将指定的弹性 IP 地址与 EC2-Classic 中的指定实例相关联。

```
C:\> Register-EC2Address -InstanceId i-12345678 -PublicIp 203.0.113.17
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[AssociateAddress](#)中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions."""

    def __init__(self, ec2_resource, elastic_ip=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
 resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param elastic_ip: A Boto3 VpcAddress object. This is a high-level object
 that
                           wraps Elastic IP actions.
        """
        self.ec2_resource = ec2_resource
        self.elastic_ip = elastic_ip

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource)


    def associate(self, instance):
```

```
        """
        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 instance: A Boto3 Instance object. This is a high-level object
that wraps
                         Amazon EC2 instance actions.
        :return: A response that contains the ID of the association.
        """
        if self.elastic_ip is None:
            logger.info("No Elastic IP to associate.")
            return

        try:
            response = self.elastic_ip.associate(InstanceId=instance.id)
        except ClientError as err:
            logger.error(
                "Couldn't associate Elastic IP %s with instance %s. Here's why:
%s: %s",
                self.elastic_ip.allocation_id,
                instance.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        return response
```

- 有关 API 的详细信息，请参阅适用AssociateAddress于 Python 的AWS SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
# 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 assocation 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,
  )
  return response.association_id
rescue StandardError => e
  puts "Error associating Elastic IP address with instance: #{e.message}"
  return "Error"
```

```
end
```

- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考AssociateAddress中的。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用AssociateAddress于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AssociateDhcpOptions`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AssociateDhcpOptions`。

CLI

AWS CLI

将 DHCP 选项集与您的 VPC 关联

此示例将指定的 DHCP 选项集与指定的 VPC 关联起来。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 associate-dhcp-options --dhcp-options-id dopt-d9070ebb --vpc-id vpc-
a01106c2
```

将默认 DHCP 选项集与您的 VPC 相关联

此示例将默认 DHCP 选项集与指定的 VPC 关联起来。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 associate-dhcp-options --dhcp-options-id default --vpc-id vpc-a01106c2
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考AssociateDhcpOptions中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的 DHCP 选项集与指定的 VPC 关联起来。

```
Register-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d -VpcId vpc-12345678
```

示例 2：此示例将默认 DHCP 选项集与指定的 VPC 关联起来。

```
Register-EC2DhcpOption -DhcpOptionsId default -VpcId vpc-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考AssociateDhcpOptions中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

AssociateRouteTable与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 AssociateRouteTable。

CLI

AWS CLI

将路由表与子网关联

此示例将指定的路由表与指定的子网关联。

命令:

```
aws ec2 associate-route-table --route-table-id rtb-22574640 --subnet-id
 subnet-9d4a7b6c
```

输出：

```
{
    "AssociationId": "rtbassoc-781d0d1a"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考AssociateRouteTable中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的路由表与指定的子网关联。

```
Register-EC2RouteTable -RouteTableId rtb-1a2b3c4d -SubnetId subnet-1a2b3c4d
```

输出：

```
rtbassoc-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考AssociateRouteTable中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AttachInternetGateway`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AttachInternetGateway`。

CLI

AWS CLI

将互联网网关连接到您的 VPC

以下`attach-internet-gateway`示例将指定的互联网网关连接到特定 VPC。

```
aws ec2 attach-internet-gateway \
    --internet-gateway-id igw-0d0fb496b3EXAMPLE \
    --vpc-id vpc-0a60eb65b4EXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon VPC 用户指南》中的互联网网关。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考AttachInternetGateway中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的互联网网关连接到指定的 VPC。

```
Add-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

示例 2：此示例创建一个 VPC 和一个互联网网关，然后将互联网网关连接到 VPC。

```
$vpc = New-EC2Vpc -CidrBlock 10.0.0.0/16
New-EC2InternetGateway | Add-EC2InternetGateway -VpcId $vpc.VpcId
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考AttachInternetGateway中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AttachNetworkInterface`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AttachNetworkInterface`。

CLI

AWS CLI

示例 1：将网络接口连接到实例

以下`attach-network-interface`示例将指定的网络接口连接到指定的实例。

```
aws ec2 attach-network-interface \
    --network-interface-id eni-0dc56a8d4640ad10a \
    --instance-id i-1234567890abcdef0 \
    --device-index 1
```

输出：

```
{
    "AttachmentId": "eni-attach-01a8fc87363f07cf9"
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的弹性网络接口。

示例 2：将网络接口连接到带有多张网卡的实例

以下`attach-network-interface`示例将指定的网络接口连接到指定的实例和网卡。

```
aws ec2 attach-network-interface \
    --network-interface-id eni-07483b1897541ad83 \
    --instance-id i-01234567890abcdef \
    --network-card-index 1 \
    --device-index 1
```

输出：

```
{
```

```
    "AttachmentId": "eni-attach-0fbd7ee87a88cd06c"
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的弹性网络接口。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考AttachNetworkInterface中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的网络接口连接到指定的实例。

```
Add-EC2NetworkInterface -NetworkInterfaceId eni-12345678 -InstanceId i-1a2b3c4d -
DeviceIndex 1
```

输出：

```
eni-attach-1a2b3c4d
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考AttachNetworkInterface中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AttachVolume`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AttachVolume`。

CLI

AWS CLI

将卷连接到实例

此示例命令将卷 (`vol-1234567890abcdef0`) 附加到实例 (`i-01474ef662b89480`) 上`/dev/sdf`。

命令:

```
aws ec2 attach-volume --volume-id vol-1234567890abcdef0 --instance-id
 i-01474ef662b89480 --device /dev/sdf
```

输出：

```
{
    "AttachTime": "YYYY-MM-DDTHH:MM:SS.000Z",
    "InstanceId": "i-01474ef662b89480",
    "VolumeId": "vol-1234567890abcdef0",
    "State": "attaching",
    "Device": "/dev/sdf"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考AttachVolume中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的卷连接到指定的实例，并使用指定的设备名称将其公开。

```
Add-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

输出：

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : attaching
VolumeId            : vol-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考AttachVolume中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AttachVpnGateway`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AttachVpnGateway`。

CLI

AWS CLI

将虚拟私有网关连接到您的 VPC

以下attach-vpn-gateway示例将指定的虚拟私有网关连接到指定的 VPC。

```
aws ec2 attach-vpn-gateway \
    --vpn-gateway-id vgw-9a4cacf3 \
    --vpc-id vpc-a01106c2
```

输出：

```
{
    "VpcAttachment": {
        "State": "attaching",
        "VpcId": "vpc-a01106c2"
    }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考AttachVpnGateway中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的虚拟私有网关连接到指定的 VPC。

```
Add-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

输出：

```
State        VpcId
-----        -----
attaching    vpc-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考AttachVpnGateway中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

AuthorizeSecurityGroupEgress与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 AuthorizeSecurityGroupEgress。

CLI

AWS CLI

添加允许出站流量到达特定地址范围的规则

此示例命令添加了一条规则，用于授予对 TCP 端口 80 上指定地址范围的访问权限。

命令（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}]'
```

命令（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}]
```

添加允许出站流量进入特定安全组的规则

此示例命令添加了一条规则，用于授予对 TCP 端口 80 上指定安全组的访问权限。

命令（Linux）：

```
aws ec2 authorize-security-group-egress --group-id sg-1a2b3c4d --ip-permissions
 IpProtocol=tcp,FromPort=80,ToPort=80,UserIdGroupPairs='[{GroupId=sg-4b51a32f}]'
```

命令（Windows）：

```
aws ec2 authorize-security-group-egress --group-id sg-1a2b3c4d --ip-permissions
 IpProtocol=tcp,FromPort=80,ToPort=80,UserIdGroupPairs=[{GroupId=sg-4b51a32f}]
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考AuthorizeSecurityGroupEgress中的。

PowerShell

用于 PowerShell

示例 1：此示例为 EC2-VPC 的指定安全组定义了出口规则。该规则授予对 TCP 端口 80 上指定 IP 地址范围的访问权限。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80";
 IpRanges="203.0.113.0/24" }
Grant-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

示例 2：在 PowerShell 版本 2 中，必须使用 New-Object 来创建对象。 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
```

示例 3：此示例在 TCP 端口 80 上授予对指定源安全组的访问权限。

```
$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 } )
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考AuthorizeSecurityGroupEgress中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AuthorizeSecurityGroupIngress`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AuthorizeSecurityGroupIngress`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用实例

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

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

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

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考[AuthorizeSecurityGroupIngress](#)中的。

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
###############################################################################
# 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 getopts 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
}
```

本示例中使用的实用程序函数。

```
################################################################################
# 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
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[AuthorizeSecurityGroupIngress](#)中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::IpRange ip_range;
    ip_range.SetCidrIp("0.0.0.0/0");

    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);

    const Aws::EC2::Model::AuthorizeSecurityGroupIngressOutcome authorizeOutcome
=
```

```
            ec2Client.AuthorizeSecurityGroupIngress(authorizeRequest);

    if (!authorizeOutcome.IsSuccess()) {
        std::cerr << "Failed to set ingress policy for security group " <<
                  groupName << ":" << authorizeOutcome.GetError().GetMessage() <<
                  std::endl;
        return false;
    }

    std::cout << "Successfully added ingress policy to security group " <<
              groupName << std::endl;
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考AuthorizeSecurityGroupIngress中的。

CLI

AWS CLI

示例 1：添加允许入站 SSH 流量的规则

以下 authorize-security-group-ingress 示例将添加一个规则，该规则允许入站流量通过 TCP 端口 22（SSH）。

```
aws ec2 authorize-security-group-ingress \
    --group-id sg-1234567890abcdef0 \
    --protocol tcp \
    --port 22 \
    --cidr 203.0.113.0/24
```

输出：

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-01afa97ef3e1bedfc",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "tcp",
```

```
            "FromPort": 22,
            "ToPort": 22,
            "CidrIpv4": "203.0.113.0/24"
        }
    ]
}
```

示例 2：添加允许来自其他安全组的入站 HTTP 流量的规则

以下 authorize-security-group-ingress 示例将添加一个规则，该规则允许源安全组通过 TCP 端口 80 进行入站访问sg-1a2b3c4d。源组必须在同一个 VPC 中，或者在对等 VPC 中（需要 VPC 对等连接）。允许的传入流量基于与源安全组相关联实例的私有 IP 地址（而不是公有 IP 或弹性 IP 地址）。

```
aws ec2 authorize-security-group-ingress \
    --group-id sg-1234567890abcdef0 \
    --protocol tcp \
    --port 80 \
    --source-group sg-1a2b3c4d
```

输出：

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-01f4be99110f638a7",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "tcp",
            "FromPort": 80,
            "ToPort": 80,
            "ReferencedGroupInfo": {
                "GroupId": "sg-1a2b3c4d",
                "UserId": "123456789012"
            }
        }
    ]
}
```

示例 3：在同一个调用中添加多个规则

以下 authorize-security-group-ingress 示例使用 ip-permissions 参数添加两个入站规则，一个允许在 TCP 端口 3389（RDP）上进行入站访问，另一个启用 ping/ICMP。

aws ec2 authorize-security-group-ingress --group-id sg-1234567890abcdef0--ip-permissions =tcp，=3389，= FromPort 3389，= “[{=172.31.0.0/16} IpProtocol]” =icmp，=-1，=-1，= “[{ToPort=172.31.0.0/16}]” IpRanges CidrIp IpProtocol FromPort ToPort IpRanges CidrIp

输出：

```
{
    "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"
        }
    ]
}
```

示例 4：为 ICMP 流量添加规则

以下 authorize-security-group-ingress 示例使用 ip-permissions 参数添加一个入站规则，该规则允许来自任何地方的 ICMP 消息 Destination Unreachable: Fragmentation Needed and Don't Fragment was Set（类型 3，代码 4）。

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

输出：

```
{
    "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"
        }
    ]
}
```

示例 5：为 IPv6 流量添加规则

以下 `authorize-security-group-ingress` 示例使用 `ip-permissions` 参数添加一个入站规则，该规则允许从 IPv6 范围 `2001:db8:1234:1a00::/64` 进行 SSH 访问（端口 22）。

aws ec2 authorize-security-group-ingress --group-id sg-1234567890abcdef0--ip-permissions =tcp，=22，=22，Ipv6Ranges= “[{6 IpProtocol =2001: db 8:1234:1 a00:: /64}]” FromPort ToPort CidrIpv

输出：

```
{
    "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"
        }
    ]
```

```
}
```

示例 6：为 ICMPv6 流量添加规则

以下 `authorize-security-group-ingress` 示例使用 `ip-permissions` 参数添加一个入站规则，该规则允许来自任何地方的 ICMPv6 流量。

aws ec2 authorize-security-group-ingress --group-id sg-1234567890abcdef0--ip-permissions =icmpv6，Ipv6Ranges= “[{6=:: /0}]” IpProtocol CidrIpv

输出：

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-04b612d9363ab6327",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "icmpv6",
            "FromPort": -1,
            "ToPort": -1,
            "CidrIpv6": "::/0"
        }
    ]
}
```

示例 7：添加带有描述的规则

以下 `authorize-security-group-ingress` 示例使用 `ip-permissions` 参数添加一个入站规则，该规则允许来自指定 IPv4 地址范围的 RDP 流量。该规则包含描述，可帮助以后识别。

aws ec2 authorize-security-group-ingress --group-id sg-1234567890abcdef0--ip-permissions =tcp，=3389，=3389，= “[{=203.0.113.0/24，FromPortdescription IpProtocol ='RDP 从纽约办公室访问'}]” ToPort IpRanges CidrIp

输出：

```
{
```

```
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "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"
        }
    ]
}
```

示例 8：添加使用前缀列表的入站规则

以下 `authorize-security-group-ingress` 示例使用 `ip-permissions` 参数添加一个入站规则，该规则允许指定前缀列表中 CIDR 范围的所有流量。

aws ec2 authorize-security-group-ingress --group-id sg-04a351bfe432d4e71--ip-permissions =all，= “[{=pl-002dc3ec097de1514}]]” IpProtocol PrefixListIds PrefixListId

输出：

```
{
    "Return": true,
    "SecurityGroupRules": [
        {
            "SecurityGroupRuleId": "sgr-09c74b32f677c6c7c",
            "GroupId": "sg-1234567890abcdef0",
            "GroupOwnerId": "123456789012",
            "IsEgress": false,
            "IpProtocol": "-1",
            "FromPort": -1,
            "ToPort": -1,
            "PrefixListId": "pl-0721453c7ac4ec009"
        }
    ]
}
```

有关更多信息，请参阅 Amazon VPC 用户指南中的安全组。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考AuthorizeSecurityGroupIngress中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
   public static String createSecurityGroup(Ec2Client ec2, String groupName,
String groupDesc, String vpcId,
           String myIpAddress) {
       try {
           CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
                   .groupName(groupName)
                   .description(groupDesc)
                   .vpcId(vpcId)
                   .build();

           CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
           IpRange ipRange = IpRange.builder()
                   .cidrIp(myIpAddress + "/0")
                   .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();

            ec2.authorizeSecurityGroupIngress(authRequest);
            System.out.println("Successfully added ingress policy to security
group " + groupName);
            return resp.groupId();

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[AuthorizeSecurityGroupIngress](#)中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#)中查找完整实例，了解如何进行设置和运行。

```
import { AuthorizeSecurityGroupIngressCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

// Grant permissions for a single IP address to ssh into instances
// within the provided security group.
export const main = async () => {
  const command = new AuthorizeSecurityGroupIngressCommand({
```

```
    // Replace with a security group ID from the AWS console or
    // the DescribeSecurityGroupsCommand.
    GroupId: "SECURITY_GROUP_ID",
    IpPermissions: [
      {
        IpProtocol: "tcp",
        FromPort: 22,
        ToPort: 22,
        // Replace 0.0.0.0 with 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: "0.0.0.0/32" }],
      },
    ],
  });

  try {
    const { SecurityGroupRules } = await client.send(command);
    console.log(JSON.stringify(SecurityGroupRules, null, 2));
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考[AuthorizeSecurityGroupIngress](#)中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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
    }
```

```
}
```

- 有关 API 的详细信息，请参阅适用AuthorizeSecurityGroupIngress于 K otlin 的AWS SDK API 参考。

PowerShell

用于 PowerShell

示例 1：此示例定义了 EC2-VPC 安全组的入口规则。这些规则授予对 SSH（端口 22）和 RDC（端口 3389）的特定 IP 地址的访问权限。请注意，您必须使用安全组 ID 而不是安全组名称来识别 EC2-VPC 的安全组。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$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 )
```

示例 2：在 PowerShell 版本 2 中，必须使用 New-Object 来创建对象。 IpPermission

```
$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 )
```

示例 3：此示例为 EC2-Classic 的安全组定义了入口规则。这些规则授予对 SSH（端口 22）和 RDC（端口 3389）的特定 IP 地址的访问权限。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$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 )
```

示例 4：在 PowerShell 版本 2 中，必须使用 New-Object 来创建对象。 IpPermission

```
$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 )
```

示例 5：此示例授予从指定源安全组 (sg-1a2b3c4d) 的 TCP 端口 8081 访问指定安全组 (sg-12345678) 的权限。

```
$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 } )
```

示例 6：此示例将 CIDR 5.5.5/32 添加到安全组 sg-1234abcd 的入口规则中，用于 TCP 端口 22 流量，并附有描述。

```
$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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[AuthorizeSecurityGroupIngress](#)中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_resource, security_group=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
 resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param security_group: A Boto3 SecurityGroup object. This is a high-level
 object
                               that wraps security group actions.
        """
        self.ec2_resource = ec2_resource
        self.security_group = security_group

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource)
```

```
   def authorize_ingress(self, ssh_ingress_ip):
       """
       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.
       """
       if self.security_group is None:
           logger.info("No security group to update.")
           return

       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.security_group.authorize_ingress(
               IpPermissions=ip_permissions
           )
       except ClientError as err:
           logger.error(
               "Couldn't authorize inbound rules for %s. Here's why: %s: %s",
               self.security_group.id,
               err.response["Error"]["Code"],
               err.response["Error"]["Message"],
           )
           raise
       else:
           return response
```

- 有关 API 的详细信息，请参阅适用AuthorizeSecurityGroupIngress于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

CancelCapacityReservation与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CancelCapacityReservation。

CLI

AWS CLI

取消容量预留

以下cancel-capacity-reservation示例取消了指定的容量预留。

```
aws ec2 cancel-capacity-reservation \
    --capacity-reservation-id cr-1234abcd56EXAMPLE
```

输出：

```
{
    "Return": true
}
```

有关更多信息，请参阅适用于 Linux 实例的 Amazon 弹性计算云用户指南中的取消容量预留。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CancelCapacityReservation中的。

PowerShell

用于 PowerShell

示例 1：此示例取消了容量预留 cr-0c1f2345db6f7cdba

```
Remove-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CancelCapacityReservation中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CancelImportTask`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CancelImportTask`。

CLI

AWS CLI

取消导入任务

以下`cancel-import-task`示例取消了指定的导入映像任务。

```
aws ec2 cancel-import-task \
    --import-task-id import-ami-1234567890abcdef0
```

输出：

```
{
    "ImportTaskId": "import-ami-1234567890abcdef0",
    "PreviousState": "active",
    "State": "deleting"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CancelImportTask中的。

PowerShell

用于 PowerShell

示例 1：此示例取消了指定的导入任务（快照或图像导入）。如果需要，可以使用**-CancelReason**参数提供理由。

```
Stop-EC2ImportTask -ImportTaskId import-ami-abcdefgh
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CancelImportTask中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

CancelSpotFleetRequests与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CancelSpotFleetRequests。

CLI

AWS CLI

示例 1：取消竞价型队列请求并终止关联的实例

以下cancel-spot-fleet-requests示例取消竞价型队列请求并终止相关的按需实例和竞价型实例。

```
aws ec2 cancel-spot-fleet-requests \
    --spot-fleet-request-ids sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE \
    --terminate-instances
```

输出：

```
{
    "SuccessfulFleetRequests": [
        {
            "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
            "CurrentSpotFleetRequestState": "cancelled_terminating",
```

```
            "PreviousSpotFleetRequestState": "active"
        }
    ],
    "UnsuccessfulFleetRequests": []
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的取消竞价型队列请求。

示例 2：在不终止关联实例的情况下取消竞价型队列请求

以下cancel-spot-fleet-requests示例在不终止关联的按需实例和竞价型实例的情况下取消竞价型队列请求。

```
aws ec2 cancel-spot-fleet-requests \
    --spot-fleet-request-ids sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE \
    --no-terminate-instances
```

输出：

```
{
    "SuccessfulFleetRequests": [
        {
            "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE",
            "CurrentSpotFleetRequestState": "cancelled_running",
            "PreviousSpotFleetRequestState": "active"
        }
    ],
    "UnsuccessfulFleetRequests": []
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的取消竞价型队列请求。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CancelSpotFleetRequests中的。

PowerShell

用于 PowerShell

示例 1：此示例取消了指定的竞价型队列请求并终止了关联的竞价型实例。

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $true
```

示例 2：此示例在不终止关联竞价型实例的情况下取消了指定的竞价型队列请求。

```
Stop-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TerminateInstance $false
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[CancelSpotFleetRequests](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CancelSpotInstanceRequests`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CancelSpotInstanceRequests`。

CLI

AWS CLI

取消竞价型实例请求

此示例命令取消竞价型实例请求。

命令:

```
aws ec2 cancel-spot-instance-requests --spot-instance-request-ids sir-08b93456
```

输出：

```
{
    "CancelledSpotInstanceRequests": [
        {
            "State": "cancelled",
            "SpotInstanceRequestId": "sir-08b93456"
        }
    ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CancelSpotInstanceRequests中的。

PowerShell

用于 PowerShell

示例 1：此示例取消了指定的竞价型实例请求。

```
Stop-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

输出：

```
SpotInstanceRequestId    State
---------------------    -----
sir-12345678             cancelled
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CancelSpotInstanceRequests中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ConfirmProductInstance`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ConfirmProductInstance`。

CLI

AWS CLI

确认产品实例

此示例确定指定的产品代码是否与指定的实例相关联。

命令:

```
aws ec2 confirm-product-instance --product-code 774F4FF8 --instance-id
 i-1234567890abcdef0
```

输出：

```
{
  "OwnerId": "123456789012"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ConfirmProductInstance中的。

PowerShell

用于 PowerShell

示例 1：此示例确定指定的产品代码是否与指定的实例相关联。

```
Confirm-EC2ProductInstance -ProductCode 774F4FF8 -InstanceId i-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ConfirmProductInstance中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

CopyImage与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CopyImage。

CLI

AWS CLI

示例 1：将 AMI 复制到另一个区域

以下copy-image示例命令将指定的 AMI 从该us-west-2区域复制到该us-east-1区域并添加简短描述。

```
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."
```

输出：

```
{
    "ImageId": "ami-0123456789abcdefg"
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的复制 AMI。

示例 2：将 AMI 复制到另一个区域并加密备份快照

以下copy-image命令将指定的 AMI 从该us-west-2区域复制到当前区域，并使用指定的 KMS 密钥对备份快照进行加密。

```
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
```

输出：

```
{
    "ImageId": "ami-0123456789abcdefg"
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的复制 AMI。

示例 3：在复制 AMI 时包含用户定义的 AMI 标签

在复制 AMI 时，以下copy-image命令使用--copy-image-tags参数复制用户定义的 AMI 标签。

```
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
```

输出：

```
{
    "ImageId": "ami-0123456789abcdefg"
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的复制 AMI。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CopyImage中的。

PowerShell

用于 PowerShell

示例 1：此示例将 “欧洲（爱尔兰）” 区域中的指定 AMI 复制到 “美国西部（俄勒冈）” 区域。如果未指定-Region，则使用当前默认区域作为目标区域。

```
Copy-EC2Image -SourceRegion eu-west-1 -SourceImageId ami-12345678 -Region us-
west-2 -Name "Copy of ami-12345678"
```

输出：

```
ami-87654321
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CopyImage中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

CopySnapshot与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CopySnapshot。

CLI

AWS CLI

示例 1：将快照复制到另一个区域

以下copy-snapshot示例命令将指定的快照从us-west-2区域复制到该us-east-1区域并添加简短描述。

```
aws ec2 copy-snapshot \
    --region us-east-1 \
    --source-region us-west-2 \
    --source-snapshot-id snap-066877671789bd71b \
    --description "This is my copied snapshot."
```

输出：

```
{
    "SnapshotId": "snap-066877671789bd71b"
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的复制 Amazon EBS 快照。

示例 2：复制未加密的快照并加密新快照

以下copy-snapshot命令将指定的未加密快照从该us-west-2区域复制到当前区域，并使用指定的 KMS 密钥对新快照进行加密。

```
aws ec2 copy-snapshot \
    --source-region us-west-2 \
    --source-snapshot-id snap-066877671789bd71b \
    --encrypted \
    --kms-key-id alias/my-kms-key
```

输出：

```
{
    "SnapshotId": "snap-066877671789bd71b"
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的复制 Amazon EBS 快照。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CopySnapshot中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的快照从欧洲（爱尔兰）地区复制到美国西部（俄勒冈）区域。

```
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678 -Region
 us-west-2
```

示例 2：如果您设置了默认区域并省略了 Region 参数，则默认目标区域为默认区域。

```
Set-DefaultAWSRegion us-west-2
Copy-EC2Snapshot -SourceRegion eu-west-1 -SourceSnapshotId snap-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CopySnapshot中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

CreateCapacityReservation与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreateCapacityReservation。

CLI

AWS CLI

示例 1：创建容量预留

以下create-capacity-reservation示例在eu-west-1a可用区创建容量预留，您可以在其中启动三个运行 Linux/Unix 操作系统的t2.medium实例。默认情况下，容量预留是在开放实例匹配条件下创建的，不支持临时存储，在您手动取消之前，容量预留将保持活动状态。

```
aws ec2 create-capacity-reservation \
    --availability-zone eu-west-1a \
    --instance-type t2.medium \
    --instance-platform Linux/UNIX \
    --instance-count 3
```

输出：

```
{
    "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"
    }
}
```

示例 2：创建在指定日期/时间自动结束的容量预留

以下create-capacity-reservation示例在eu-west-1a可用区创建容量预留，您可以在其中启动三个运行 Linux/Unix 操作系统的m5.large实例。此容量预留将于 2019 年 8 月 31 日 23:59:59 自动结束。

```
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
```

输出：

```
{
    "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"
    }
}
```

示例 3：创建仅接受目标实例启动的容量预留

以下create-capacity-reservation示例创建了仅接受目标实例启动的容量预留。

```
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
```

输出：

```
{
    "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"
    }
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的创建容量预留。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateCapacityReservation中的。

PowerShell

用于 PowerShell

示例 1：此示例使用指定属性创建新的容量预留

```
Add-EC2CapacityReservation -InstanceType m4.xlarge -InstanceCount 2 -
AvailabilityZone eu-west-1b -EbsOptimized True -InstancePlatform Windows
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateCapacityReservation中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateCustomerGateway`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateCustomerGateway`。

CLI

AWS CLI

创建客户网关

此示例为其外部接口创建了一个具有指定 IP 地址的客户网关。

命令:

```
aws ec2 create-customer-gateway --type ipsec.1 --public-ip 12.1.2.3 --bgp-asn
 65534
```

输出：

```
{
    "CustomerGateway": {
        "CustomerGatewayId": "cgw-0e11f167",
        "IpAddress": "12.1.2.3",
        "State": "available",
        "Type": "ipsec.1",
        "BgpAsn": "65534"
    }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateCustomerGateway中的。

PowerShell

用于 PowerShell

示例 1：此示例创建了指定的客户网关。

```
New-EC2CustomerGateway -Type ipsec.1 -PublicIp 203.0.113.12 -BgpAsn 65534
```

输出：

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
State             : available
Tags              : {}
Type              : ipsec.1
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateCustomerGateway中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateDhcpOptions`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateDhcpOptions`。

CLI

AWS CLI

创建一组 DHCP 选项

以下`create-dhcp-options`示例创建了一组 DHCP 选项，用于指定域名、域名服务器和 NetBIOS 节点类型。

```
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"
```

输出：

```
{
    "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"
    }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[CreateDhcpOptions](#)中的。

PowerShell

用于 PowerShell

示例 1：此示例创建指定的 DHCP 选项集。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$options = @( @{Key="domain-name";Values=@("abc.local")}, @{Key="domain-name-
servers";Values=@("10.0.0.101","10.0.0.102")})
New-EC2DhcpOption -DhcpConfiguration $options
```

输出：

```
DhcpConfigurations                    DhcpOptionsId    Tags
------------------                    -------------    ----
{domain-name, domain-name-servers}    dopt-1a2b3c4d    {}
```

示例 2：在 PowerShell 版本 2 中，必须使用 New-Object 来创建每个 DHCP 选项。

```
$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)
```

输出：

```
DhcpConfigurations                    DhcpOptionsId    Tags
------------------                    -------------    ----
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateDhcpOptions中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

CreateFlowLogs与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreateFlowLogs。

CLI

AWS CLI

示例 1：创建流日志

以下create-flow-logs示例创建了一个流日志，用于捕获指定网络接口的所有被拒绝的流量。使用指定 IAM 角色中的权限将流 CloudWatch 日志传输到 Logs 中的日志组。

```
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
```

输出：

```
{
```

```
    "ClientToken": "so0eNA2uSHUNlHI0S2cJ305GuIX1CezaRdGtexample",
    "FlowLogIds": [
        "fl-12345678901234567"
    ],
    "Unsuccessful": []
}
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的 VPC 流日志。

示例 2：使用自定义格式创建流日志

以下create-flow-logs示例创建了一个流日志，用于捕获指定 VPC 的所有流量并将流日志传输到 Amazon S3 存储桶。--log-format 参数指定流日志记录的自定义格式。要在 Windows 上运行此命令，请将单引号 (') 更改为双引号 (")。

```
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}'
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的 VPC 流日志。

示例 3：创建最大聚合间隔为一分钟的流日志

以下create-flow-logs示例创建了一个流日志，用于捕获指定 VPC 的所有流量并将流日志传输到 Amazon S3 存储桶。该--max-aggregation-interval参数将最大聚合间隔指定为 60 秒（1 分钟）。

```
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
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的 VPC 流日志。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateFlowLogs中的。

PowerShell

用于 PowerShell

示例 1：此示例使用 “管理员” 角色的权限将子网子网-1d234567 的 EC2 流日志创建到 cloud-watch-log 名为 “subnet1-log” 的所有 “拒绝” 流量

```
New-EC2FlowLog -ResourceId "subnet-1d234567" -LogDestinationType cloud-watch-
logs -LogGroupName subnet1-log -TrafficType "REJECT" -ResourceType Subnet -
DeliverLogsPermissionArn "arn:aws:iam::98765432109:role/Admin"
```

输出：

```
ClientToken                                     FlowLogIds             Unsuccessful
-----------                                     ----------             ------------
m1VN2cxP3iB4qo//VUKl5EU6cF7gQLOxcqNefvjeTGw= {fl-012fc34eed5678c9d} {}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateFlowLogs中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateImage`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateImage`。

CLI

AWS CLI

示例 1：从 Amazon EBS 支持的实例创建 AMI

以下`create-image`示例从指定实例创建 AMI。

```
aws ec2 create-image \
    --instance-id i-1234567890abcdef0 \
    --name "My server" \
```

```
    --description "An AMI for my server"
```

输出：

```
{
    "ImageId": "ami-abcdef01234567890"
}
```

有关为 AMI 指定块储存设备映射的更多信息，请参阅 Amazon EC2 用户指南中的为 AMI 指定块储存设备映射。

示例 2：在不重启的情况下从 Amazon EBS 支持的实例创建 AMI

以下create-image示例创建了一个 AMI 并设置了--no-reboot 参数，这样在创建映像之前就不会重启实例。

```
aws ec2 create-image \
    --instance-id i-1234567890abcdef0 \
    --name "My server" \
    --no-reboot
```

输出：

```
{
    "ImageId": "ami-abcdef01234567890"
}
```

有关为 AMI 指定块储存设备映射的更多信息，请参阅 Amazon EC2 用户指南中的为 AMI 指定块储存设备映射。

示例 3：在创建时标记 AMI 和快照

以下create-image示例创建了一个 AMI，并使用相同的标签为 AMI 和快照添加了标签 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}]"
```

输出：

```
{
    "ImageId": "ami-abcdef01234567890"
}
```

有关在创建时为资源添加标签的更多信息，请参阅 Amazon EC2 用户指南中的在资源创建时添加标签。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateImage中的。

PowerShell

用于 PowerShell

示例 1：此示例从指定实例创建具有指定名称和描述的 AMI。Amazon EC2 尝试在创建映像之前彻底关闭实例，并在完成后重新启动实例。

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI"
```

示例 2：此示例从指定实例创建具有指定名称和描述的 AMI。Amazon EC2 无需关闭和重启实例即可创建映像；因此，无法保证所创建映像的文件系统的完整性。

```
New-EC2Image -InstanceId i-12345678 -Name "my-web-server" -Description "My web
 server AMI" -NoReboot $true
```

示例 3：此示例创建了一个包含三个卷的 AMI。第一个卷基于 Amazon EBS 快照。第二个卷是一个空的 100 GiB 亚马逊 EBS 卷。第三个卷是实例存储卷。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$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"})
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateImage中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateInstanceExportTask`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateInstanceExportTask`。

CLI

AWS CLI

导出实例

此示例命令创建一个任务，用于将实例 i-1234567890abcdef0 导出到 Amazon S3 存储桶 myexportbucket。

命令:

```
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
```

输出：

```
{
    "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"
    }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateInstanceExportTask中的。

PowerShell

用于 PowerShell

示例 1：此示例将已停止的实例作为虚拟硬盘 (VHD) 导出到 S3 存储桶**testbucket-export-instances-2019**。**i-0800b00a00EXAMPLE**目标环境是**Microsoft**，添加区域参数是因为实例位于该区域，而用户的默认**us-east-1** AWS 区域不是 us-east-1。要获取导出任务的状态，请复制此命令结果中的**ExportTaskId**值，然后运行 **Get-EC2ExportTask -ExportTaskId export_task_ID_from_results.**

```
New-EC2InstanceExportTask -InstanceId i-0800b00a00EXAMPLE -
ExportToS3Task_DiskImageFormat VHD -ExportToS3Task_S3Bucket "testbucket-export-
instances-2019" -TargetEnvironment Microsoft -Region us-east-1
```

输出：

```
Description           :
ExportTaskId          : export-i-077c73108aEXAMPLE
ExportToS3Task        : Amazon.EC2.Model.ExportToS3Task
InstanceExportDetails : Amazon.EC2.Model.InstanceExportDetails
State                 : active
StatusMessage         :
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateInstanceExportTask中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

CreateInternetGateway与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreateInternetGateway。

CLI

AWS CLI

创建互联网网关

以下create-internet-gateway示例使用标签创建互联网网关Name=my-igw。

```
aws ec2 create-internet-gateway \
    --tag-specifications ResourceType=internet-gateway,Tags=[{Key=Name,Value=my-
igw}]
```

输出：

```
{
    "InternetGateway": {
        "Attachments": [],
        "InternetGatewayId": "igw-0d0fb496b3994d755",
        "OwnerId": "123456789012",
        "Tags": [
            {
                "Key": "Name",
                "Value": "my-igw"
            }
        ]
    }
}
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的互联网网关。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateInternetGateway中的。

PowerShell

用于 PowerShell

示例 1：此示例创建互联网网关。

```
New-EC2InternetGateway
```

输出：

```
Attachments    InternetGatewayId    Tags
-----------    -----------------    ----
{}             igw-1a2b3c4d         {}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateInternetGateway中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateKeyPair`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateKeyPair`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用实例

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
/// <summary>
/// Create an Amazon EC2 key pair.
/// </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)
{
    var request = new CreateKeyPairRequest
    {
        KeyName = keyPairName,
    };

    var response = await _amazonEC2.CreateKeyPairAsync(request);

    if (response.HttpStatusCode == HttpStatusCode.OK)
    {
        var kp = response.KeyPair;
        return kp;
    }
```

```
        else
        {
            Console.WriteLine("Could not create key pair.");
            return null;
        }
    }

    /// <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;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考CreateKeyPair中的。

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
##############################################################################
# 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 getopts 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
}
```

本示例中使用的实用程序函数。

```
###############################################################################
# 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
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateKeyPair中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
   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:" <<
                 outcome.GetError().GetMessage() << std::endl;
   }
   else {
       std::cout << "Successfully created key pair named " <<
                 keyPairName << std::endl;
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考CreateKeyPair中的。

CLI

AWS CLI

创建密钥对

本示例将创建一个名为 MyKeyPair 的密钥对。

命令:

```
aws ec2 create-key-pair --key-name MyKeyPair
```

输出是私有密钥和密钥指纹的 ASCII 版本。需要将密钥保存到文件中。

有关更多信息，请参阅《AWS 命令行界面用户指南》中的“使用密钥对”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateKeyPair中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
   public static void createKeyPair(Ec2Client ec2, String keyName, String
fileName) {
       try {
           CreateKeyPairRequest request = CreateKeyPairRequest.builder()
                   .keyName(keyName)
                   .build();

           CreateKeyPairResponse response = ec2.createKeyPair(request);
           String content = response.keyMaterial();
           BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
           writer.write(content);
           writer.close();
           System.out.println("Successfully created key pair named " + keyName);

       } catch (Ec2Exception | IOException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateKeyPair中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { CreateKeyPairCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Create a key pair in Amazon EC2.
    const { KeyMaterial, KeyName } = await client.send(
      // A unique name for the key pair. Up to 255 ASCII characters.
      new CreateKeyPairCommand({ KeyName: "KEY_PAIR_NAME" }),
    );
    // This logs your private key. Be sure to save it.
    console.log(KeyName);
    console.log(KeyMaterial);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考CreateKeyPair中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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}")
    }
}
```

- 有关 API 的详细信息，请参阅适用CreateKeyPair于 K otlin 的AWS SDK API 参考。

PowerShell

用于 PowerShell

示例 1：此示例创建一个密钥对，并在具有指定名称的文件中捕获经过 PEM 编码的 RSA 私钥。使用时 PowerShell，必须将编码设置为 ascii 才能生成有效的密钥。有关更多信息，请参阅 AWS 命令行界面用户指南中的创建、显示和删除 Amazon EC2 密钥对 (https://docs.aws.amazon.com/cli/latest/userguide/ cli-services-ec 2-keypairs.html)。

```
(New-EC2KeyPair -KeyName "my-key-pair").KeyMaterial | Out-File -Encoding ascii -
FilePath C:\path\my-key-pair.pem
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateKeyPair中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class KeyPairWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair
 actions."""

    def __init__(self, ec2_resource, key_file_dir, key_pair=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
 resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param key_file_dir: The folder where the private key information is
 stored.
                             This should be a secure folder.
        :param key_pair: A Boto3 KeyPair object. This is a high-level object that
                         wraps key pair actions.
        """
        self.ec2_resource = ec2_resource
        self.key_pair = key_pair
        self.key_file_path = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource, tempfile.TemporaryDirectory())


    def create(self, key_name):
        """
        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 Boto3 KeyPair object that represents the newly created key
 pair.
        """
        try:
            self.key_pair = self.ec2_resource.create_key_pair(KeyName=key_name)
            self.key_file_path = os.path.join(
                self.key_file_dir.name, f"{self.key_pair.name}.pem"
```

```
            )
            with open(self.key_file_path, "w") as key_file:
                key_file.write(self.key_pair.key_material)
        except ClientError as err:
            logger.error(
                "Couldn't create key %s. Here's why: %s: %s",
                key_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.key_pair
```

- 有关 API 的详细信息，请参阅适用CreateKeyPair于 Python 的AWS SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
# 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}'."
  return true
rescue Aws::EC2::Errors::InvalidKeyPairDuplicate
  puts "Error creating key pair: a key pair named '#{key_pair_name}' " \
    "already exists."
  return false
rescue StandardError => e
  puts "Error creating key pair or saving private key file: #{e.message}"
  return 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
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)
  return true
rescue StandardError => e
  puts "Error deleting key pair: #{e.message}"
  return 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__
```

- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考[CreateKeyPair](#)中的。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用CreateKeyPair于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateLaunchTemplate`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateLaunchTemplate`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 构建和管理弹性服务

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
   /// <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)
   {
       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;

    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考CreateLaunchTemplate中的。

CLI

AWS CLI

示例 1：创建启动模板

以下 create-launch-template 示例创建一个启动模板，该模板指定启动实例的子网，为实例分配公有 IP 地址和 IPv6 地址，并为实例创建标签。

```
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"}]}]}'
```

输出：

```
{
    "LaunchTemplate": {
        "LatestVersionNumber": 1,
```

```
        "LaunchTemplateId": "lt-01238c059e3466abc",
        "LaunchTemplateName": "TemplateForWebServer",
        "DefaultVersionNumber": 1,
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "CreateTime": "2019-01-27T09:13:24.000Z"
    }
}
```

有关更多信息，请参阅《Amazon Elastic Compute Cloud 用户指南》中的“从启动模板启动实例”。有关引用 JSON 格式参数的信息，请参阅《AWS 命令行界面用户指南》中的“引用字符串”。

示例 2：为 Amazon EC2 Auto Scaling 创建启动模板

以下 `create-launch-template` 示例创建一个具有多个标签和块设备映射的启动模板，以在实例启动时指定一个附加 EBS 卷。为 `Groups` 指定一个值，该值与 VPC 的安全组对应，自动扩缩组会将实例启动到该安全组中。指定 VPC 和子网作为自动扩缩组的属性。

```
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
```

输出：

```
{
    "LaunchTemplate": {
        "LatestVersionNumber": 1,
        "LaunchTemplateId": "lt-0123c79c33a54e0abc",
        "LaunchTemplateName": "TemplateForAutoScaling",
        "DefaultVersionNumber": 1,
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "CreateTime": "2019-04-30T18:16:06.000Z"
    }
```

```
}
```

有关更多信息，请参阅《Amazon EC2 Auto Scaling 用户指南》中的“为自动扩缩组创建启动模板”。有关引用 JSON 格式参数的信息，请参阅《AWS 命令行界面用户指南》中的“引用字符串”。

示例 3：创建指定对 EBS 卷进行加密的启动模板

以下 `create-launch-template` 示例创建一个启动模板，其中包括根据未加密的快照创建的加密 EBS 卷。同时还会在创建过程中标记卷。如果默认情况下禁用了加密，则必须指定以下示例中所示的 `"Encrypted"` 选项。如果使用 `"KmsKeyId"` 选项来指定客户托管的 CMK，则即使默认情况下启用了加密，也必须指定 `"Encrypted"` 选项。

```
aws ec2 create-launch-template \
  --launch-template-name TemplateForEncryption \
  --launch-template-data file://config.json
```

`config.json` 的内容：

```
{
    "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"
```

```
                }
            ]
        }
    ]
}
```

输出：

```
{
    "LaunchTemplate": {
        "LatestVersionNumber": 1,
        "LaunchTemplateId": "lt-0d5bd51bcf8530abc",
        "LaunchTemplateName": "TemplateForEncryption",
        "DefaultVersionNumber": 1,
        "CreatedBy": "arn:aws:iam::123456789012:user/Bob",
        "CreateTime": "2020-01-07T19:08:36.000Z"
    }
}
```

有关更多信息，请参阅《Amazon Elastic Compute Cloud 用户指南》中的“从快照恢复 Amazon EBS 卷且默认情况下加密”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateLaunchTemplate中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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,
        },
      }),
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考[CreateLaunchTemplate](#)中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

此示例创建一个启动模板，其中包括向实例授予特定权限的实例配置文件以及启动后在实例上运行的用户数据 Bash 脚本。

```
class AutoScaler:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
```

```
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :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
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"
        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"
        self.key_pair_name = f"{resource_prefix}-key-pair"


    def create_template(self, server_startup_script_file, instance_policy_file):
        """
        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:
            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,
            )
            with open(server_startup_script_file) as file:
                start_server_script = file.read()
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]
            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(
                "Created launch template %s for AMI %s on %s.",
                self.launch_template_name,
                ami_id,
                self.inst_type,
            )
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
```

```
                == "InvalidLaunchTemplateName.AlreadyExistsException"
            ):
                log.info(
                    "Launch template %s already exists, nothing to do.",
                    self.launch_template_name,
                )
            else:
                raise AutoScalerError(
                    f"Couldn't create launch template
{self.launch_template_name}: {err}."
                )
        return template
```

- 有关 API 的详细信息，请参阅适用CreateLaunchTemplate于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateNetworkAcl`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateNetworkAcl`。

CLI

AWS CLI

创建网络 ACL

此示例为指定的 VPC 创建网络 ACL。

命令:

```
aws ec2 create-network-acl --vpc-id vpc-a01106c2
```

输出：

```
{
```

```
    "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
    }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateNetworkAcl中的。

PowerShell

用于 PowerShell

示例 1：此示例为指定的 VPC 创建网络 ACL。

```
New-EC2NetworkAcl -VpcId vpc-12345678
```

输出：

```
Associations : {}
Entries      : {Amazon.EC2.Model.NetworkAclEntry,
 Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
```

```
Tags         : {}
VpcId        : vpc-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateNetworkAcl中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateNetworkAclEntry`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateNetworkAclEntry`。

CLI

AWS CLI

创建网络 ACL 条目

此示例为指定的网络 ACL 创建了一个条目。该规则允许从 UDP 端口 53 (DNS) 上的任何 IPv4 地址 (0.0.0.0/0) 进入任何关联子网的入口流量。如果命令成功，则不返回任何输出。

命令:

```
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
```

此示例为指定的网络 ACL 创建了一条规则，该规则允许通过 TCP 端口 80 (HTTP) 来自任何 IPv6 地址 (:: /0) 的入口流量。

命令:

```
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
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateNetworkAclEntry中的。

PowerShell

用于 PowerShell

示例 1：此示例为指定的网络 ACL 创建了一个条目。该规则允许从 UDP 端口 53 (DNS) 上的任何地方 (0.0.0.0/0) 进入任何关联子网的入站流量。

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateNetworkAclEntry中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateNetworkInterface`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateNetworkInterface`。

CLI

AWS CLI

示例 1：为网络接口指定 IPv4 地址

以下`create-network-interface`示例使用指定的主 IPv4 地址为指定子网创建网络接口。

```
aws ec2 create-network-interface \
    --subnet-id subnet-00a24d0d67acf6333 \
    --description "my network interface" \
    --groups sg-09dfba7ed20cda78b \
    --private-ip-address 10.0.8.17
```

输出：

```
{
    "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"
    }
}
```

示例 2：创建具有 IPv4 地址和 IPv6 地址的网络接口

以下`create-network-interface`示例使用由 Amazon EC2 选择的 IPv4 地址和 IPv6 地址为指定子网创建网络接口。

```
aws ec2 create-network-interface \
    --subnet-id subnet-00a24d0d67acf6333 \
    --description "my dual stack network interface" \
    --ipv6-address-count 1 \
    --groups sg-09dfba7ed20cda78b
```

输出：

```
{
    "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"
    }
}
```

示例 3：使用连接跟踪配置选项创建网络接口

以下create-network-interface示例创建了一个网络接口并配置了空闲连接跟踪超时。

```
aws ec2 create-network-interface \
    --subnet-id subnet-00a24d0d67acf6333 \
    --groups sg-02e57dbcfe0331c1b \
    --connection-tracking-specification TcpEstablishedTimeout=86400,UdpTimeout=60
```

输出：

```
{
    "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"
    }
```

```
}
```

示例 4：创建弹性结构适配器

以下create-network-interface示例创建了 EFA。

```
aws ec2 create-network-interface \
    --interface-type efa \
    --subnet-id subnet-00a24d0d67acf6333 \
    --description "my efa" \
    --groups sg-02e57dbcfe0331c1b
```

输出：

```
{
    "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"
    }
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的弹性网络接口。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateNetworkInterface中的。

PowerShell

用于 PowerShell

示例 1：此示例创建指定的网络接口。

```
New-EC2NetworkInterface -SubnetId subnet-1a2b3c4d -Description "my network
 interface" -Group sg-12345678 -PrivateIpAddress 10.0.0.17
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateNetworkInterface中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

CreatePlacementGroup与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreatePlacementGroup。

CLI

AWS CLI

创建置放群组

此示例命令使用指定名称创建置放群组。

命令:

```
aws ec2 create-placement-group --group-name my-cluster --strategy cluster
```

创建分区置放群组

此示例命令创建一个名HDFS-Group-A为五个分区的分区置放组。

命令:

```
aws ec2 create-placement-group --group-name HDFS-Group-A --strategy partition --
partition-count 5
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreatePlacementGroup中的。

PowerShell

用于 PowerShell

示例 1：此示例使用指定名称创建置放群组。

```
New-EC2PlacementGroup -GroupName my-placement-group -Strategy cluster
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreatePlacementGroup中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateRoute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateRoute`。

CLI

AWS CLI

创建路线

此示例为指定的路由表创建路由。该路由匹配所有 IPv4 流量 (`0.0.0.0/0`)，并将其路由到指定的互联网网关。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 create-route --route-table-id rtb-22574640 --destination-cidr-block
 0.0.0.0/0 --gateway-id igw-c0a643a9
```

此示例命令在路由表 rtb-g8ff4ea2 中创建路由。该路由匹配 IPv4 CIDR 块 10.0.0.0/16 的流量，并将其路由到 VPC 对等连接 pcx-111aaa22。通过此路由，可以将流量定向到 VPC 对等连接中的对等 VPC。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 create-route --route-table-id rtb-g8ff4ea2 --destination-cidr-block
 10.0.0.0/16 --vpc-peering-connection-id pcx-1a2b3c4d
```

此示例在指定的路由表中创建一个匹配所有 IPv6 流量 (`::/0`) 的路由，并将其路由到指定的仅限出口 Internet 网关。

命令:

```
aws ec2 create-route --route-table-id rtb-dce620b8 --destination-ipv6-cidr-
block ::/0 --egress-only-internet-gateway-id eigw-01eadbd45ecd7943f
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateRoute中的。

PowerShell

用于 PowerShell

示例 1：此示例为指定路由表创建指定路由。该路由匹配所有流量并将其发送到指定的 Internet 网关。

```
New-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0 -
GatewayId igw-1a2b3c4d
```

输出：

```
True
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateRoute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateRouteTable`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateRouteTable`。

CLI

AWS CLI

创建路由表

本示例为指定的 VPC 创建路由表。

命令:

```
aws ec2 create-route-table --vpc-id vpc-a01106c2
```

输出：

```
{
    "RouteTable": {
```

```
        "Associations": [],
        "RouteTableId": "rtb-22574640",
        "VpcId": "vpc-a01106c2",
        "PropagatingVgws": [],
        "Tags": [],
        "Routes": [
            {
                "GatewayId": "local",
                "DestinationCidrBlock": "10.0.0.0/16",
                "State": "active"
            }
        ]
    }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateRouteTable中的。

PowerShell

用于 PowerShell

示例 1：此示例为指定 VPC 创建路由表。

```
New-EC2RouteTable -VpcId vpc-12345678
```

输出：

```
Associations    : {}
PropagatingVgws : {}
Routes          : {}
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateRouteTable中的。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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
      }
    ]
  )
  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}'."
  return 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."
  return 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__
```

- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考CreateRouteTable中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateSecurityGroup`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateSecurityGroup`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用实例

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    /// <summary>
    /// Create an Amazon EC2 security group.
    /// </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)
```

```
    {
        var response = await _amazonEC2.CreateSecurityGroupAsync(
            new CreateSecurityGroupRequest(groupName, groupDescription));

        return response.GroupId;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考CreateSecurityGroup中的。

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
###############################################################################
# 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
}
```

本示例中使用的实用程序函数。

```
###############################################################################
# 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
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateSecurityGroup中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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;
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考[CreateSecurityGroup](#)中的。

CLI

AWS CLI

为 EC2-Classic 创建安全组

本示例将创建一个名为 MySecurityGroup 的安全组。

命令:

```
aws ec2 create-security-group --group-name MySecurityGroup --description "My
 security group"
```

输出：

```
{
    "GroupId": "sg-903004f8"
}
```

为 EC2-VPC 创建安全组

本示例为指定的 VPC 创建名为 MySecurityGroup 的安全组。

命令:

```
aws ec2 create-security-group --group-name MySecurityGroup --description "My
 security group" --vpc-id vpc-1a2b3c4d
```

输出：

```
{
```

```
    "GroupId": "sg-903004f8"
}
```

有关更多信息，请参阅《AWS 命令行界面用户指南》中的“使用安全组”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateSecurityGroup中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
   public static String createSecurityGroup(Ec2Client ec2, String groupName,
String groupDesc, String vpcId,
           String myIpAddress) {
       try {
           CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
                   .groupName(groupName)
                   .description(groupDesc)
                   .vpcId(vpcId)
                   .build();

           CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
           IpRange ipRange = IpRange.builder()
                   .cidrIp(myIpAddress + "/0")
                   .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();

            ec2.authorizeSecurityGroupIngress(authRequest);
            System.out.println("Successfully added ingress policy to security
group " + groupName);
            return resp.groupId();

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考CreateSecurityGroup中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { CreateSecurityGroupCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

export const main = async () => {
```

```
  const command = new CreateSecurityGroupCommand({
    // Up to 255 characters in length. Cannot start with sg-.
    GroupName: "SECURITY_GROUP_NAME",
    // Up to 255 characters in length.
    Description: "DESCRIPTION",
  });

  try {
    const { GroupId } = await client.send(command);
    console.log(GroupId);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考CreateSecurityGroup中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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
    }
}
```

- 有关 API 的详细信息，请参阅适用CreateSecurityGroup于 K otlin 的AWS SDK API 参考。

PowerShell

用于 PowerShell

示例 1：此示例为指定的 VPC 创建安全组。

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security
 group" -VpcId vpc-12345678
```

输出：

```
sg-12345678
```

示例 2：此示例为 EC2-Classic 创建了一个安全组。

```
New-EC2SecurityGroup -GroupName my-security-group -Description "my security
 group"
```

输出：

```
sg-45678901
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[CreateSecurityGroup](#)中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_resource, security_group=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
 resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
```

```
        :param security_group: A Boto3 SecurityGroup object. This is a high-level
object
                               that wraps security group actions.
        """
        self.ec2_resource = ec2_resource
        self.security_group = security_group

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource)


    def create(self, group_name, group_description):
        """
        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: A Boto3 SecurityGroup object that represents the newly created
security group.
        """
        try:
            self.security_group = self.ec2_resource.create_security_group(
                GroupName=group_name, Description=group_description
            )
        except ClientError as err:
            logger.error(
                "Couldn't create security group %s. Here's why: %s: %s",
                group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.security_group
```

- 有关 API 的详细信息，请参阅适用CreateSecurityGroup于 Python 的AWS SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
# 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}'."
  return security_group.group_id
rescue StandardError => e
  puts "Error creating security group: #{e.message}"
  return "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}'."
  return true
rescue StandardError => e
  puts "Error adding inbound rule to security group: #{e.message}"
  return false
end

# Displays information about a security group's IP permissions set in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - A security group with inbound rules, outbound rules, or both.
#
# @param p [Aws::EC2::Types::IpPermission] The IP permissions set.
# @example
#   ec2_client = Aws::EC2::Client.new(region: 'us-west-2')
#   response = ec2_client.describe_security_groups
#   unless sg.ip_permissions.empty?
#     describe_security_group_permissions(
```

```
#       response.security_groups[0].ip_permissions[0]
#     )
#   end
def describe_security_group_permissions(perm)
  print "  Protocol: #{perm.ip_protocol == '-1' ? 'All' : perm.ip_protocol}"

  unless perm.from_port.nil?
    if perm.from_port == "-1" || perm.from_port == -1
      print ", From: All"
    else
      print ", From: #{perm.from_port}"
    end
  end

  unless perm.to_port.nil?
    if perm.to_port == "-1" || perm.to_port == -1
      print ", To: All"
    else
      print ", To: #{perm.to_port}"
    end
  end

  if perm.key?(:ipv_6_ranges) && perm.ipv_6_ranges.count.positive?
    print ", CIDR IPv6: #{perm.ipv_6_ranges[0].cidr_ipv_6}"
  end

  if perm.key?(:ip_ranges) && perm.ip_ranges.count.positive?
    print ", CIDR IPv4: #{perm.ip_ranges[0].cidr_ip}"
  end

  print "\n"
end

# Displays information about available security groups in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @example
#   describe_security_groups(Aws::EC2::Client.new(region: 'us-west-2'))
def describe_security_groups(ec2_client)
  response = ec2_client.describe_security_groups

  if response.security_groups.count.positive?
    response.security_groups.each do |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}"

      if sg.tags.count.positive?
        puts "Tags:"
        sg.tags.each do |tag|
          puts "  Key: #{tag.key}, Value: #{tag.value}"
        end
      end

      unless sg.ip_permissions.empty?
        puts "Inbound rules:" if sg.ip_permissions.count.positive?
        sg.ip_permissions.each do |p|
          describe_security_group_permissions(p)
        end
      end

      unless sg.ip_permissions_egress.empty?
        puts "Outbound rules:" if sg.ip_permissions.count.positive?
        sg.ip_permissions_egress.each do |p|
          describe_security_group_permissions(p)
        end
      end
    end
  else
    puts "No security groups found."
  end
rescue StandardError => e
  puts "Error getting information about security groups: #{e.message}"
end

# Deletes 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 to delete.
# @return [Boolean] true if the security group was deleted; otherwise, false.
# @example
#   exit 1 unless security_group_deleted?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'sg-030a858e078f1b9EX'
#   )
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}'."
  return true
rescue StandardError => e
  puts "Error deleting security group: #{e.message}"
  return false
end

# Example usage:
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 = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    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"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    group_name = "my-security-group"
    description = "This is my security group."
```

```
  vpc_id = "vpc-6713dfEX"
  ip_protocol_http = "tcp"
  from_port_http = "80"
  to_port_http = "80"
  cidr_ip_range_http = "0.0.0.0/0"
  ip_protocol_ssh = "tcp"
  from_port_ssh = "22"
  to_port_ssh = "22"
  cidr_ip_range_ssh = "0.0.0.0/0"
  # 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
  group_name = ARGV[0]
  description = ARGV[1]
  vpc_id = ARGV[2]
  ip_protocol_http = ARGV[3]
  from_port_http = ARGV[4]
  to_port_http = ARGV[5]
  cidr_ip_range_http = ARGV[6]
  ip_protocol_ssh = ARGV[7]
  from_port_ssh = ARGV[8]
  to_port_ssh = ARGV[9]
  cidr_ip_range_ssh = ARGV[10]
  region = ARGV[11]
end

security_group_id = ""
security_group_exists = false
ec2_client = Aws::EC2::Client.new(region: region)

puts "Attempting to create security group..."
security_group_id = create_security_group(
  ec2_client,
  group_name,
  description,
  vpc_id
)
if security_group_id == "Error"
  puts "Could not create security group. Skipping this step."
else
  security_group_exists = true
end
```

```
  if security_group_exists
    puts "Attempting to add inbound rules to security group..."
    unless security_group_ingress_authorized?(
      ec2_client,
      security_group_id,
      ip_protocol_http,
      from_port_http,
      to_port_http,
      cidr_ip_range_http
    )
      puts "Could not add inbound HTTP rule to security group. " \
        "Skipping this step."
    end

    unless security_group_ingress_authorized?(
      ec2_client,
      security_group_id,
      ip_protocol_ssh,
      from_port_ssh,
      to_port_ssh,
      cidr_ip_range_ssh
    )
      puts "Could not add inbound SSH rule to security group. " \
        "Skipping this step."
    end
  end

  puts "\nInformation about available security groups:"
  describe_security_groups(ec2_client)

  if security_group_exists
    puts "\nAttempting to delete security group..."
    unless security_group_deleted?(ec2_client, security_group_id)
      puts "Could not delete security group. You must delete it yourself."
    end
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考CreateSecurityGroup中的。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用CreateSecurityGroup于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateSnapshot`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateSnapshot`。

CLI

AWS CLI

创建快照

此示例命令创建卷的快照，卷 ID 为，vol-1234567890abcdef0并附上用于识别快照的简短描述。

命令:

```
aws ec2 create-snapshot --volume-id vol-1234567890abcdef0 --description "This is
 my root volume snapshot"
```

输出：

```
{
    "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"
}
```

创建带有标签的快照

此示例命令创建快照并应用两个标签：purpose=prod 和 costcenter=123。

命令:

```
aws ec2 create-snapshot --volume-id vol-1234567890abcdef0
 --description 'Prod backup' --tag-specifications
 'ResourceType=snapshot,Tags=[{Key=purpose,Value=prod},
{Key=costcenter,Value=123}]'
```

输出：

```
{
    "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"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateSnapshot中的。

PowerShell

用于 PowerShell

示例 1：此示例创建指定卷的快照。

```
New-EC2Snapshot -VolumeId vol-12345678 -Description "This is a test"
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateSnapshot中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateSpotDatafeedSubscription`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateSpotDatafeedSubscription`。

CLI

AWS CLI

创建竞价型实例数据源

以下`create-spot-datafeed-subscription`示例创建了竞价型实例数据馈送。

```
aws ec2 create-spot-datafeed-subscription \
    --bucket my-bucket \
    --prefix spot-data-feed
```

输出：

```
{
    "SpotDatafeedSubscription": {
        "Bucket": "my-bucket",
        "OwnerId": "123456789012",
        "Prefix": "spot-data-feed",
        "State": "Active"
    }
}
```

数据源存储在您指定的 Amazon S3 存储桶中。此数据馈送的文件名采用以下格式。

```
my-bucket.s3.amazonaws.com/spot-data-feed/123456789012.YYYY-MM-DD-
HH.n.abcd1234.gz
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的竞价型实例数据源。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateSpotDatafeedSubscription中的。

PowerShell

用于 PowerShell

示例 1：此示例创建竞价型实例数据馈送。

```
New-EC2SpotDatafeedSubscription -Bucket my-s3-bucket -Prefix spotdata
```

输出：

```
Bucket  : my-s3-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateSpotDatafeedSubscription中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateSubnet`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateSubnet`。

CLI

AWS CLI

示例 1：创建仅具有 IPv4 CIDR 块的子网

以下 `create-subnet` 示例在指定的 VPC 中创建具有指定 IPv4 CIDR 块的子网。

```
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}]
```

输出：

```
{
    "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"
    }
}
```

示例 2：创建同时具有 IPv4 和 IPv6 CIDR 块的子网

以下 `create-subnet` 示例在指定的 VPC 中，创建同时具有指定 IPv4 和 IPv6 CIDR 块的子网。

```
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}]
```

输出：

```
{
    "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"
    }
}
```

示例 3：创建仅具有 IPv6 CIDR 块的子网

以下 `create-subnet` 示例在指定的 VPC 中创建具有指定 IPv6 CIDR 块的子网。

```
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}]
```

输出：

```
{
    "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"
    }
}
```

有关更多信息，请参阅 Amazon VPC 用户指南中的 VPC 和子网。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateSubnet中的。

PowerShell

用于 PowerShell

示例 1：此示例使用指定 CIDR 创建子网。

```
New-EC2Subnet -VpcId vpc-12345678 -CidrBlock 10.0.0.0/24
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[CreateSubnet](#)中的。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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_vlue [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}'."
  return true
rescue StandardError => e
  puts "Error creating or tagging subnet: #{e.message}"
  return 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__
```

- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考CreateSubnet中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

CreateTags与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreateTags。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::Tag nameTag;
    nameTag.SetKey("Name");
    nameTag.SetValue(instanceName);

    Aws::EC2::Model::CreateTagsRequest createRequest;
    createRequest.AddResources(instanceID);
    createRequest.AddTags(nameTag);
```

```
    Aws::EC2::Model::CreateTagsOutcome createOutcome = ec2Client.CreateTags(
            createRequest);
    if (!createOutcome.IsSuccess()) {
        std::cerr << "Failed to tag ec2 instance " << instanceID <<
                  " with name " << instanceName << ":" <<
                  createOutcome.GetError().GetMessage() << std::endl;
        return false;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考CreateTags中的。

CLI

AWS CLI

示例 1：为资源添加标签

以下 create-tags 示例将标签 Stack=production 添加到指定的映像，或者覆盖 AMI 的现有标签（其中标签键为 Stack）。

```
aws ec2 create-tags \
    --resources ami-1234567890abcdef0 \
    --tags Key=Stack,Value=production
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的主题标题。

示例 2：为多个资源添加标签

以下 create-tags 示例为 AMI 和实例添加（或覆盖）两个标签。其中一个标签有一个键（webserver），但没有值（值设置为空字符串）。另一个标签有一个键（stack）和一个值（Production）。

```
aws ec2 create-tags \
    --resources ami-1a2b3c4d i-1234567890abcdef0 \
    --tags Key=webserver,Value=   Key=stack,Value=Production
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的主题标题。

示例 3：添加包含特殊字符的标签

以下 create-tags 示例为实例添加标签 [Group]=test。方括号（[和]）是特殊字符，必须对其进行转义。以下示例还使用适用于每个环境的行延续字符。

如果使用的是 Windows，请用双引号（"）将具有特殊字符的元素引起来，然后在每个双引号字符前面添加反斜杠（\），如下所示：

```
aws ec2 create-tags ^
    --resources i-1234567890abcdef0 ^
    --tags Key=\"[Group]\",Value=test
```

如果您使用的是 Windows PowerShell，请在元素中使用双引号 (“) 将带有特殊字符的值括起来，在每个双引号字符前面加一个反斜杠 (\)，然后用单引号 (') 将整个键和值结构括起来，如下所示：

```
aws ec2 create-tags `
    --resources i-1234567890abcdef0 `
    --tags 'Key=\"[Group]\",Value=test'
```

如果使用的是 Linux 或 OS X，请使用双引号（"）将具有特殊字符的元素引起来，然后使用单引号（'）将整个键和值结构引起来，如下所示：

```
aws ec2 create-tags \
    --resources i-1234567890abcdef0 \
    --tags 'Key="[Group]",Value=test'
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的主题标题。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateTags中的。

PowerShell

用于 PowerShell

示例 1：此示例向指定资源添加单个标签。标签键为 “myTag”，标签值为 “myTagValue”。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
New-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag"; Value="myTagValue" }
```

示例 2：此示例更新或向指定资源添加指定标签。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
New-EC2Tag -Resource i-12345678 -Tag @( @{ Key="myTag"; Value="newTagValue" },
 @{ Key="test"; Value="anotherTagValue" } )
```

示例 3：在 PowerShell 版本 2 中，必须使用 New-Object 为标签参数创建标签。

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

New-EC2Tag -Resource i-12345678 -Tag $tag
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[CreateTags](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateVolume`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateVolume`。

CLI

AWS CLI

创建空的通用固态硬盘 (gp2) 卷

以下`create-volume`示例在指定的可用区创建一个 80 GiB 的通用型 SSD (gp2) 卷。请注意，当前区域必须是`us-east-1`，或者您可以添加`--region`参数来为命令指定区域。

```
aws ec2 create-volume \
    --volume-type gp2 \
    --size 80 \
    --availability-zone us-east-1a
```

输出：

```
{
    "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
}
```

如果您未指定卷类型，则默认卷类型为gp2。

```
aws ec2 create-volume \
    --size 80 \
    --availability-zone us-east-1a
```

示例 2：使用快照创建预配置 IOPS 固态硬盘 (io1) 卷

以下create-volume示例使用指定的快照在指定可用区中创建一个预配置 IOPS SSD (io1) 卷，该卷具有 1000 个预配置 IOPS SSD (io1)。

```
aws ec2 create-volume \
    --volume-type io1 \
    --iops 1000 \
    --snapshot-id snap-066877671789bd71b \
    --availability-zone us-east-1a
```

输出：

```
{
    "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
}
```

示例 3：创建加密卷

以下create-volume示例使用用于 EBS 加密的默认 CMK 创建加密卷。如果默认情况下加密处于禁用状态，则必须按以下方式指定--encrypted参数。

```
aws ec2 create-volume \
    --size 80 \
    --encrypted \
    --availability-zone us-east-1a
```

输出：

```
{
    "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
}
```

如果默认启用了加密，则以下示例命令将创建加密卷，即使没有--encrypted参数也是如此。

```
aws ec2 create-volume \
    --size 80 \
    --availability-zone us-east-1a
```

如果您使用--kms-key-id参数来指定客户管理的 CMK，则即使默认启用了加密，也必须指定该--encrypted参数。

```
aws ec2 create-volume \
    --volume-type gp2 \
    --size 80 \
    --encrypted \
    --kms-key-id 0ea3fef3-80a7-4778-9d8c-1c0c6EXAMPLE \
    --availability-zone us-east-1a
```

示例 4：创建带有标签的卷

以下create-volume示例创建了一个卷并添加了两个标签。

```
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}]'
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateVolume中的。

PowerShell

用于 PowerShell

示例 1：此示例创建了指定的卷。

```
New-EC2Volume -Size 50 -AvailabilityZone us-west-2a -VolumeType gp2
```

输出：

```
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
```

示例 2：此示例请求创建卷并应用带有堆栈密钥和生产值的标签。

```
$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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateVolume中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateVpc`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateVpc`。

CLI

AWS CLI

示例 1：创建 VPC

以下 `create-vpc` 示例创建具有指定 IPv4 CIDR 块和名称标签的 VPC。

```
aws ec2 create-vpc \
    --cidr-block 10.0.0.0/16 \
    --tag-specifications ResourceType=vpc,Tags=[{Key=Name,Value=MyVpc}]
```

输出：

```
{
    "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"
            }
        ]
    }
}
```

示例 2：创建具有专用租赁的 VPC

以下 create-vpc 示例创建具有指定 IPv4 CIDR 块和专用租赁的 VPC。

```
aws ec2 create-vpc \
    --cidr-block 10.0.0.0/16 \
    --instance-tenancy dedicated
```

输出：

```
{
    "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
    }
}
```

示例 3：创建具有 IPv6 CIDR 块的 VPC

以下 `create-vpc` 示例创建具有 Amazon 提供的 IPv6 CIDR 块的 VPC。

```
aws ec2 create-vpc \
    --cidr-block 10.0.0.0/16 \
    --amazon-provided-ipv6-cidr-block
```

输出：

```
{
    "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
    }
}
```

示例 4：从 IPAM 池中创建具有 CIDR 的 VPC

以下 `create-vpc` 示例从 Amazon VPC IP 地址管理器（IPAM）池，创建具有 CIDR 的 VPC。

Linux 和 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"}]
```

输出：

```
{
    "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"
            }
        ]
    }
}
```

有关更多信息，请参阅《Amazon VPC IPAM 用户指南》中的创建使用 IPAM 池 CIDR 的 VPC。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateVpc中的。

PowerShell

用于 PowerShell

示例 1：此示例使用指定 CIDR 创建一个 VPC。Amazon VPC 还会为 VPC 创建以下内容：默认 DHCP 选项集、主路由表和默认网络 ACL。

```
New-EC2VPC -CidrBlock 10.0.0.0/16
```

输出：

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : pending
Tags            : {}
VpcId           : vpc-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateVpc中的。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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}'."
  return true
rescue StandardError => e
  puts "#{e.message}"
  return 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__
```

- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考CreateVpc中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateVpcEndpoint`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateVpcEndpoint`。

CLI

AWS CLI

示例 1：创建网关终端节点

以下`create-vpc-endpoint`示例`us-east-1`在该区域的 VPC `vpc-1a2b3c4d` 和 Amazon S3 之间创建网关 VPC 终端节点，并将路由表`rtb-11aa22bb`与该终端节点关联起来。

```
aws ec2 create-vpc-endpoint \
    --vpc-id vpc-1a2b3c4d \
    --service-name com.amazonaws.us-east-1.s3 \
    --route-table-ids rtb-11aa22bb
```

输出：

```
{
    "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"
    }
}
```

有关更多信息，请参阅AWS PrivateLink 指南中的创建网关终端节点。

示例 2：创建接口终端节点

以下create-vpc-endpoint示例us-east-1在该区域的 VPC vpc-1a2b3c4d 和 Amazon S3 之间创建了一个接口 VPC 终端节点。该命令在子网中创建终端节点subnet-1a2b3c4d，将其与安全组关联sg-1a2b3c4d，并添加密钥为 “服务”、值为 “S3” 的标签。

```
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}]
```

输出：

```
{
    "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"
    }
}
```

有关更多信息，请参阅《用户指南》中的创建接口端点 AWS PrivateLink。

示例 3：创建 Gateway Load Balancer 终端节点

以下create-vpc-endpoint示例在 VPC vpc-111122223333aabbc 和之间创建网关负载均衡器终端节点，以及使用网关负载均衡器配置的服务。

```
aws ec2 create-vpc-endpoint \
    --service-name com.amazonaws.vpce.us-east-1.vpce-svc-123123a1c43abc123 \
```

```
    --vpc-endpoint-type GatewayLoadBalancer \
    --vpc-id vpc-111122223333aabbc \
    --subnet-ids subnet-0011aabbcc2233445
```

输出：

```
{
    "VpcEndpoint": {
        "VpcEndpointId": "vpce-aabbaabbaabbaabba",
        "VpcEndpointType": "GatewayLoadBalancer",
        "VpcId": "vpc-111122223333aabbc",
        "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"
    }
}
```

有关更多信息，请参阅《用户指南》中的 Gateway Load Balancer 终端节点 AWS PrivateLink。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateVpcEndpoint中的。

PowerShell

用于 PowerShell

示例 1：此示例在 VPC vpc-0fc1ff23f45b678eb 中为服务 com.amazonaws.eu-west-1.s3 创建新的 VPC 终端节点

```
New-EC2VpcEndpoint -ServiceName com.amazonaws.eu-west-1.s3 -VpcId
 vpc-0fc1ff23f45b678eb
```

输出：

```
ClientToken VpcEndpoint
```

```
----------- -----------
            Amazon.EC2.Model.VpcEndpoint
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateVpcEndpoint中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

CreateVpnConnection与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreateVpnConnection。

CLI

AWS CLI

示例 1：使用动态路由创建 VPN 连接

以下create-vpn-connection示例在指定的虚拟专用网关和指定的客户网关之间创建 VPN 连接，并将标签应用于 VPN 连接。输出包括 XML 格式的客户网关设备的配置信息。

```
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}]'
```

输出：

```
{
    "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"
            }
        ]
    }
}
```

有关更多信息，请参阅站点到站点 VPN 用户指南中的 AWS 站点到站点 VPN 的工作原理。AWS

示例 2：使用静态路由创建 VPN 连接

以下create-vpn-connection示例在指定的虚拟专用网关和指定的客户网关之间创建 VPN 连接。这些选项指定静态路由。输出包括 XML 格式的客户网关设备的配置信息。

```
aws ec2 create-vpn-connection \
    --type ipsec.1 \
    --customer-gateway-id cgw-001122334455aabbc \
    --vpn-gateway-id vgw-1a1a1a1a1a1a2b2b2 \
    --options "{\"StaticRoutesOnly\":true}"
```

输出：

```
{
    "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": []
    }
}
```

有关更多信息，请参阅站点到站点 VPN 用户指南中的 AWS 站点到站点 VPN 的工作原理。AWS

示例 3：创建 VPN 连接并指定自己的内部 CIDR 和预共享密钥

以下create-vpn-connection示例创建 VPN 连接并为每个隧道指定内部 IP 地址 CIDR 块和自定义预共享密钥。指定的值将在CustomerGatewayConfiguration信息中返回。

```
aws ec2 create-vpn-connection \
    --type ipsec.1 \
    --customer-gateway-id cgw-001122334455aabbc \
    --vpn-gateway-id vgw-1a1a1a1a1a1a2b2b2 \
    --options
 TunnelOptions='[{TunnelInsideCidr=169.254.12.0/30,PreSharedKey=ExamplePreSharedKey1},
{TunnelInsideCidr=169.254.13.0/30,PreSharedKey=ExamplePreSharedKey2}]'
```

输出：

```
{
    "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": [
                {
                    "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": []
    }
}
```

有关更多信息，请参阅站点到站点 VPN 用户指南中的 AWS 站点到站点 VPN 的工作原理。AWS

示例 4：创建支持 IPv6 流量的 VPN 连接

以下create-vpn-connection示例创建了一个 VPN 连接，该连接支持指定传输网关和指定客户网关之间的 IPv6 流量。两条隧道的隧道选项都指定了 AWS 必须启动 IKE 协商的隧道。

```
aws ec2 create-vpn-connection \
    --type ipsec.1 \
    --transit-gateway-id tgw-12312312312312312 \
    --customer-gateway-id cgw-001122334455aabbc \
    --options TunnelInsideIpVersion=ipv6,TunnelOptions=[{StartupAction=start},
{StartupAction=start}]
```

输出：

```
{
    "VpnConnection": {
        "CustomerGatewayConfiguration": "..configuration information...",
        "CustomerGatewayId": "cgw-001122334455aabbc",
```

```
        "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": []
    }
}
```

有关更多信息，请参阅站点到站点 VPN 用户指南中的 AWS 站点到站点 VPN 的工作原理。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateVpnConnection中的。

PowerShell

用于 PowerShell

示例 1：此示例在指定的虚拟专用网关和指定的客户网关之间创建 VPN 连接。输出包括您的网络管理员所需的配置信息，采用 XML 格式。

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d
```

输出：

```
CustomerGatewayConfiguration : [XML document]
CustomerGatewayId            : cgw-1a2b3c4d
Options                      :
Routes                       : {}
State                        : pending
Tags                         : {}
Type                         :
VgwTelemetry                 : {}
VpnConnectionId              : vpn-12345678
VpnGatewayId                 : vgw-1a2b3c4d
```

示例 2：此示例创建 VPN 连接并将配置捕获到具有指定名称的文件中。

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-
configuration.xml
```

示例 3：此示例使用静态路由在指定的虚拟专用网关和指定的客户网关之间创建一个 VPN 连接。

```
New-EC2VpnConnection -Type ipsec.1 -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
 vgw-1a2b3c4d -Options_StaticRoutesOnly $true
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateVpnConnection中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateVpnConnectionRoute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateVpnConnectionRoute`。

CLI

AWS CLI

为 VPN 连接创建静态路由

此示例为指定的 VPN 连接创建静态路由。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 create-vpn-connection-route --vpn-connection-id vpn-40f41529 --
destination-cidr-block 11.12.0.0/16
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateVpnConnectionRoute中的。

PowerShell

用于 PowerShell

示例 1：此示例为指定的 VPN 连接创建指定的静态路由。

```
New-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考CreateVpnConnectionRoute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateVpnGateway`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateVpnGateway`。

CLI

AWS CLI

创建虚拟专用网关

此示例创建了一个虚拟专用网关。

命令:

```
aws ec2 create-vpn-gateway --type ipsec.1
```

输出：

```
{
```

```
    "VpnGateway": {
        "AmazonSideAsn": 64512,
        "State": "available",
        "Type": "ipsec.1",
        "VpnGatewayId": "vgw-9a4cacf3",
        "VpcAttachments": []
    }
}
```

使用特定亚马逊端 ASN 创建虚拟专用网关

此示例创建了一个虚拟专用网关，并为 BGP 会话的 Amazon 端指定自治系统编号 (ASN)。

命令:

```
aws ec2 create-vpn-gateway --type ipsec.1 --amazon-side-asn 65001
```

输出：

```
{
    "VpnGateway": {
        "AmazonSideAsn": 65001,
        "State": "available",
        "Type": "ipsec.1",
        "VpnGatewayId": "vgw-9a4cacf3",
        "VpcAttachments": []
    }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考CreateVpnGateway中的。

PowerShell

用于 PowerShell

示例 1：此示例创建了指定的虚拟专用网关。

```
New-EC2VpnGateway -Type ipsec.1
```

输出：

```
AvailabilityZone :
```

```
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {}
VpnGatewayId     : vgw-1a2b3c4d
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[CreateVpnGateway](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteCustomerGateway`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteCustomerGateway`。

CLI

AWS CLI

删除客户网关

此示例删除了指定的客户网关。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-customer-gateway --customer-gateway-id cgw-0e11f167
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DeleteCustomerGateway](#)中的。

PowerShell

用于 PowerShell

示例 1：此示例删除了指定的客户网关。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

输出：

```
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"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteCustomerGateway中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteDhcpOptions`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteDhcpOptions`。

CLI

AWS CLI

删除 DHCP 选项集

此示例删除指定的 DHCP 选项集。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-dhcp-options --dhcp-options-id dopt-d9070ebb
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteDhcpOptions中的。

PowerShell

用于 PowerShell

示例 1：此示例删除指定的 DHCP 选项集。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d
```

输出：

```
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"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteDhcpOptions中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DeleteFlowLogs与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteFlowLogs。

CLI

AWS CLI

删除流日志

以下delete-flow-logs示例删除了指定的流日志。

```
aws ec2 delete-flow-logs --flow-log-id fl-11223344556677889
```

输出：

```
{
    "Unsuccessful": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteFlowLogs中的。

PowerShell

用于 PowerShell

示例 1：此示例删除了给定的 FlowLogId fl-01a2b3456a789c01

```
Remove-EC2FlowLog -FlowLogId fl-01a2b3456a789c01
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteFlowLogs中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DeleteInternetGateway与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteInternetGateway。

CLI

AWS CLI

删除互联网网关

以下delete-internet-gateway示例删除了指定的互联网网关。

```
aws ec2 delete-internet-gateway \
    --internet-gateway-id igw-0d0fb496b3EXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon VPC 用户指南》中的互联网网关。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteInternetGateway中的。

PowerShell

用于 PowerShell

示例 1：此示例删除指定的互联网网关。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

输出：

```
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"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteInternetGateway中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteKeyPair`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteKeyPair`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用实例

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考DeleteKeyPair中的。

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
##############################################################################
# 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 getopts 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 getopts "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
    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
}
```

本示例中使用的实用程序函数。

```
################################################################################
# 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
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteKeyPair中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考DeleteKeyPair中的。

CLI

AWS CLI

删除密钥对

以下`delete-key-pair`示例删除指定的 key pair。

```
aws ec2 delete-key-pair \
    --key-name my-key-pair
```

输出：

```
{
    "Return": true,
    "KeyPairId": "key-03c8d3aceb53b507"
}
```

有关更多信息，请参阅《AWS 命令行界面用户指南》中的创建和删除密钥对。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteKeyPair中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
public static void deleteKeys(Ec2Client ec2, String keyPair) {
    try {
        DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
                .keyName(keyPair)
                .build();

        ec2.deleteKeyPair(request);
        System.out.println("Successfully deleted key pair named " + keyPair);

    } catch (Ec2Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteKeyPair中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { DeleteKeyPairCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

export const main = async () => {
  const command = new DeleteKeyPairCommand({
    KeyName: "KEY_PAIR_NAME",
  });

  try {
    await client.send(command);
    console.log("Successfully deleted key pair.");
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DeleteKeyPair中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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")
    }
}
```

- 有关 API 的详细信息，请参阅适用DeleteKeyPair于 K otlin 的AWS SDK API 参考。

PowerShell

用于 PowerShell

示例 1：此示例删除指定的 key pair。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2KeyPair -KeyName my-key-pair
```

输出：

```
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"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteKeyPair中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class KeyPairWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair
 actions."""

    def __init__(self, ec2_resource, key_file_dir, key_pair=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
 resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param key_file_dir: The folder where the private key information is
 stored.
                             This should be a secure folder.
        :param key_pair: A Boto3 KeyPair object. This is a high-level object that
                         wraps key pair actions.
        """
        self.ec2_resource = ec2_resource
        self.key_pair = key_pair
        self.key_file_path = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource, tempfile.TemporaryDirectory())


    def delete(self):
        """
        Deletes a key pair.
        """
        if self.key_pair is None:
```

```
            logger.info("No key pair to delete.")
            return

        key_name = self.key_pair.name
        try:
            self.key_pair.delete()
            self.key_pair = None
        except ClientError as err:
            logger.error(
                "Couldn't delete key %s. Here's why: %s : %s",
                key_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- 有关 API 的详细信息，请参阅适用DeleteKeyPair于 Python 的AWS SDK (Boto3) API 参考。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用DeleteKeyPair于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteLaunchTemplate`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteLaunchTemplate`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 构建和管理弹性服务

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
/// <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 (AmazonClientException)
    {
        Console.WriteLine($"Unable to delete template {templateName}.");
    }
```

```
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考DeleteLaunchTemplate中的。

CLI

AWS CLI

删除启动模板

本示例将删除指定的启动模板。

命令:

```
aws ec2 delete-launch-template --launch-template-id lt-0abcd290751193123
```

输出：

```
{
  "LaunchTemplate": {
      "LatestVersionNumber": 2,
      "LaunchTemplateId": "lt-0abcd290751193123",
      "LaunchTemplateName": "TestTemplate",
      "DefaultVersionNumber": 2,
      "CreatedBy": "arn:aws:iam::123456789012:root",
      "CreateTime": "2017-11-23T16:46:25.000Z"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteLaunchTemplate中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    await client.send(
      new DeleteLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
      }),
    );
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考[DeleteLaunchTemplate](#)中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class AutoScaler:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :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
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"
        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"
        self.key_pair_name = f"{resource_prefix}-key-pair"


    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,
                )
```

```
            else:
                raise AutoScalerError(
                    f"Couldn't delete launch template
{self.launch_template_name}: {err}."
                )
```

- 有关 API 的详细信息，请参阅适用DeleteLaunchTemplate于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteNetworkAcl`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteNetworkAcl`。

CLI

AWS CLI

删除网络 ACL

此示例删除指定的网络 ACL。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-network-acl --network-acl-id acl-5fb85d36
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteNetworkAcl中的。

PowerShell

用于 PowerShell

示例 1：此示例删除指定的网络 ACL。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2NetworkAcl -NetworkAclId acl-12345678
```

输出：

```
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"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteNetworkAcl中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteNetworkAclEntry`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteNetworkAclEntry`。

CLI

AWS CLI

删除网络 ACL 条目

此示例从指定的网络 ACL 中删除了编号为 100 的入口规则。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-network-acl-entry --network-acl-id acl-5fb85d36 --ingress --rule-
number 100
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteNetworkAclEntry中的。

PowerShell

用于 PowerShell

示例 1：此示例从指定的网络 ACL 中删除指定的规则。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2NetworkAclEntry -NetworkAclId acl-12345678 -Egress $false -RuleNumber
 100
```

输出：

```
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"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteNetworkAclEntry中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteNetworkInterface`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteNetworkInterface`。

CLI

AWS CLI

删除网络接口

此示例删除了指定的网络接口。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-network-interface --network-interface-id eni-e5aa89a3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteNetworkInterface中的。

PowerShell

用于 PowerShell

示例 1：此示例删除指定的网络接口。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

输出：

```
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"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteNetworkInterface中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeletePlacementGroup`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeletePlacementGroup`。

CLI

AWS CLI

删除置放群组

此示例命令删除指定的置放群组。

命令:

```
aws ec2 delete-placement-group --group-name my-cluster
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeletePlacementGroup中的。

PowerShell

用于 PowerShell

示例 1：此示例删除了指定的置放群组。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2PlacementGroup -GroupName my-placement-group
```

输出：

```
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"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeletePlacementGroup中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteRoute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteRoute`。

CLI

AWS CLI

删除路线

此示例从指定路由表中删除指定路由。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-route --route-table-id rtb-22574640 --destination-cidr-block
 0.0.0.0/0
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteRoute中的。

PowerShell

用于 PowerShell

示例 1：此示例从指定路由表中删除指定路由。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 0.0.0.0/0
```

输出：

```
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"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteRoute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteRouteTable`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteRouteTable`。

CLI

AWS CLI

删除路由表

此示例删除了指定的路由表。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-route-table --route-table-id rtb-22574640
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteRouteTable中的。

PowerShell

用于 PowerShell

示例 1：此示例删除了指定的路由表。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

输出：

```
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"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteRouteTable中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteSecurityGroup`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteSecurityGroup`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用实例

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    /// <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)
    {
        var response = await _amazonEC2.DeleteSecurityGroupAsync(new
DeleteSecurityGroupRequest { GroupId = groupId });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考DeleteSecurityGroup中的。

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
###############################################################################
# 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 getopts 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 getopts "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
}
```

本示例中使用的实用程序函数。

```
################################################################################
# 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
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteSecurityGroup中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DeleteSecurityGroupRequest request;

    request.SetGroupId(securityGroupID);
    auto 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;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考DeleteSecurityGroup中的。

CLI

AWS CLI

[EC2-Classic] 删除安全组

本示例将删除名为 MySecurityGroup 的安全组。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-security-group --group-name MySecurityGroup
```

[EC2-VPC] 删除安全组

本示例将删除 ID 为 sg-903004f8 的安全组。请注意，您不能通过名称引用 EC2-VPC 的安全组。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-security-group --group-id sg-903004f8
```

有关更多信息，请参阅《AWS 命令行界面用户指南》中的“使用安全组”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteSecurityGroup中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
        try {
            DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
```

```
                    .groupId(groupId)
                    .build();

            ec2.deleteSecurityGroup(request);
            System.out.println("Successfully deleted security group with Id " +
 groupId);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DeleteSecurityGroup中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { DeleteSecurityGroupCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

export const main = async () => {
  const command = new DeleteSecurityGroupCommand({
    GroupId: "GROUP_ID",
  });

  try {
    await client.send(command);
    console.log("Security group deleted successfully.");
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DeleteSecurityGroup中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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")
    }
}
```

- 有关 API 的详细信息，请参阅适用DeleteSecurityGroup于 K otlin 的AWS SDK API 参考。

PowerShell

用于 PowerShell

示例 1：此示例删除了 EC2-VPC 的指定安全组。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2SecurityGroup -GroupId sg-12345678
```

输出：

```
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"):
```

示例 2：此示例删除了 EC2-Classic 的指定安全组。

```
Remove-EC2SecurityGroup -GroupName my-security-group -Force
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DeleteSecurityGroup](#)中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_resource, security_group=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
 resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param security_group: A Boto3 SecurityGroup object. This is a high-level
 object
                               that wraps security group actions.
        """
        self.ec2_resource = ec2_resource
        self.security_group = security_group
```

```
    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource)


    def delete(self):
        """
        Deletes the security group.
        """
        if self.security_group is None:
            logger.info("No security group to delete.")
            return

        group_id = self.security_group.id
        try:
            self.security_group.delete()
        except ClientError as err:
            logger.error(
                "Couldn't delete security group %s. Here's why: %s: %s",
                group_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- 有关 API 的详细信息，请参阅适用DeleteSecurityGroup于 Python 的AWS SDK (Boto3) API 参考。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用DeleteSecurityGroup于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteSnapshot`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteSnapshot`。

CLI

AWS CLI

删除快照

本示例命令将删除快照 ID 为 `snap-1234567890abcdef0` 的快照。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-snapshot --snapshot-id snap-1234567890abcdef0
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteSnapshot中的。

PowerShell

用于 PowerShell

示例 1：此示例删除指定的快照。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2Snapshot -SnapshotId snap-12345678
```

输出：

```
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"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteSnapshot中的。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
async fn delete_snapshot(client: &Client, id: &str) -> Result<(), Error> {
    client.delete_snapshot().snapshot_id(id).send().await?;

    println!("Deleted");

    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用DeleteSnapshot于 Rust 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DeleteSpotDatafeedSubscription与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteSpotDatafeedSubscription。

CLI

AWS CLI

取消竞价型实例数据源订阅

此示例命令删除该账户的竞价数据源订阅。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-spot-datafeed-subscription
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteSpotDatafeedSubscription中的。

PowerShell

用于 PowerShell

示例 1：此示例删除您的竞价型实例数据 Feed。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2SpotDatafeedSubscription
```

输出：

```
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"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteSpotDatafeedSubscription中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DeleteSubnet与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteSubnet。

CLI

AWS CLI

删除子网

此示例删除了指定的子网。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-subnet --subnet-id subnet-9d4a7b6c
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteSubnet中的。

PowerShell

用于 PowerShell

示例 1：此示例删除指定的子网。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2Subnet -SubnetId subnet-1a2b3c4d
```

输出：

```
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"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteSubnet中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DeleteTags与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteTags。

CLI

AWS CLI

示例 1：从资源中删除标签

以下delete-tags示例Stack=Test从指定图像中删除标签。当您同时指定值和密钥名称时，只有当标签的值与指定值匹配时，才会删除该标签。

```
aws ec2 delete-tags \
    --resources ami-1234567890abcdef0 \
    --tags Key=Stack,Value=Test
```

为标签指定值是可选的。以下delete-tags示例将purpose从指定实例中删除带有密钥名称的标签，而不管该标签的标签值如何。

```
aws ec2 delete-tags \
    --resources i-1234567890abcdef0 \
    --tags Key=purpose
```

如果将空字符串指定为标签值，则仅当标签的值为空字符串时，才会删除标记。以下delete-tags示例将空字符串指定为要删除的标签的标签值。

```
aws ec2 delete-tags \
    --resources i-1234567890abcdef0 \
    --tags Key=Name,Value=
```

示例 2：从多个资源中删除标签

以下delete-tags示例从实例和 AMI 中删除标签 “purpose=test``”。如前面的示例所示，您可以省略命令中的标签值。

```
aws ec2 delete-tags \
    --resources i-1234567890abcdef0 ami-1234567890abcdef0 \
    --tags Key=Purpose
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteTags中的。

PowerShell

用于 PowerShell

示例 1：此示例从指定资源中删除指定标签，无论标签值如何。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag" } -Force
```

示例 2：此示例将从指定资源中删除指定的标签，但前提是标签值匹配。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
Remove-EC2Tag -Resource i-12345678 -Tag @{ Key="myTag";Value="myTagValue" } -
Force
```

示例 3：此示例从指定资源中删除指定标签，无论标签值如何。

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

示例 4：此示例将从指定资源中删除指定的标签，但前提是标签值匹配。

```
$tag = New-Object Amazon.EC2.Model.Tag
$tag.Key = "myTag"
$tag.Value = "myTagValue"

Remove-EC2Tag -Resource i-12345678 -Tag $tag -Force
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteTags中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteVolume`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteVolume`。

CLI

AWS CLI

删除卷

此示例命令删除卷 ID 为的可用卷vol-049df61146c4d7901。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-volume --volume-id vol-049df61146c4d7901
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteVolume中的。

PowerShell

用于 PowerShell

示例 1：此示例分离指定的卷。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2Volume -VolumeId vol-12345678
```

输出：

```
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"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteVolume中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DeleteVpc与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteVpc。

CLI

AWS CLI

删除 VPC

此示例删除指定的 VPC。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-vpc --vpc-id vpc-a01106c2
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteVpc中的。

PowerShell

用于 PowerShell

示例 1：此示例删除指定的 VPC。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2Vpc -VpcId vpc-12345678
```

输出：

```
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"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteVpc中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DeleteVpnConnection与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteVpnConnection。

CLI

AWS CLI

删除 VPN 连接

此示例删除指定的 VPN 连接。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-vpn-connection --vpn-connection-id vpn-40f41529
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteVpnConnection中的。

PowerShell

用于 PowerShell

示例 1：此示例删除指定的 VPN 连接。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2VpnConnection -VpnConnectionId vpn-12345678
```

输出：

```
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"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteVpnConnection中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteVpnConnectionRoute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteVpnConnectionRoute`。

CLI

AWS CLI

从 VPN 连接中删除静态路由

此示例从指定的 VPN 连接中删除指定的静态路由。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-vpn-connection-route --vpn-connection-id vpn-40f41529 --
destination-cidr-block 11.12.0.0/16
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteVpnConnectionRoute中的。

PowerShell

用于 PowerShell

示例 1：此示例从指定的 VPN 连接中删除指定的静态路由。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2VpnConnectionRoute -VpnConnectionId vpn-12345678 -DestinationCidrBlock
 11.12.0.0/16
```

输出：

```
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"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteVpnConnectionRoute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteVpnGateway`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteVpnGateway`。

CLI

AWS CLI

删除虚拟专用网关

此示例删除了指定的虚拟专用网关。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 delete-vpn-gateway --vpn-gateway-id vgw-9a4cacf3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeleteVpnGateway中的。

PowerShell

用于 PowerShell

示例 1：此示例删除指定的虚拟专用网关。除非您还指定了 Force 参数，否则在操作继续之前，系统会提示您进行确认。

```
Remove-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

输出：

```
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"):
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeleteVpnGateway中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeregisterImage`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeregisterImage`。

CLI

AWS CLI

取消注册 AMI

此示例取消注册指定的 AMI。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 deregister-image --image-id ami-4fa54026
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DeregisterImage中的。

PowerShell

用于 PowerShell

示例 1：此示例取消注册指定的 AMI。

```
Unregister-EC2Image -ImageId ami-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DeregisterImage中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeAccountAttributes`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeAccountAttributes`。

CLI

AWS CLI

描述您 AWS 账户的所有属性

此示例描述了您的 AWS 账户的属性。

命令:

```
aws ec2 describe-account-attributes
```

输出：

```
{
    "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"
                }
            ]
        }
    ]
}
```

描述您 AWS 账户的单个属性

此示例描述了您的 AWS 账户的supported-platforms属性。

命令:

```
aws ec2 describe-account-attributes --attribute-names supported-platforms
```

输出：

```
{
    "AccountAttributes": [
        {
            "AttributeName": "supported-platforms",
            "AttributeValues": [
                {
                    "AttributeValue": "EC2"
                },
                {
                    "AttributeValue": "VPC"
                }
```

```
            ]
        }
    ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeAccountAttributes](#)中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了您是可以在该地区的 EC2-Classic 和 EC2-VPC 中启动实例，还是只能在 EC2-VPC 中启动实例。

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

输出：

```
AttributeValue
--------------
EC2
VPC
```

示例 2：此示例描述了您的默认 VPC，或者如果您在该地区没有默认 VPC，则为 “无”。

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

输出：

```
AttributeValue
--------------
vpc-12345678
```

示例 3：此示例描述了您可以运行的最大按需实例数。

```
(Get-EC2AccountAttribute -AttributeName max-instances).AttributeValues
```

输出：

```
AttributeValue
--------------
20
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DescribeAccountAttributes](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeAddresses`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeAddresses`。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#)中查找完整实例，了解如何进行设置和运行。

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DescribeAddressesRequest request;
auto 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 auto &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;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考DescribeAddresses中的。

CLI

AWS CLI

示例 1：检索所有弹性 IP 地址的详细信息

以下 `describe addresses` 示例显示有关弹性 IP 地址的详细信息。

```
aws ec2 describe-addresses
```

输出：

```
{
    "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"
        }
    ]
}
```

示例 2：检索 EC2-VPC 弹性 IP 地址的详细信息

以下 describe-addresses 示例显示，与 VPC 中的实例配合使用的弹性 IP 地址的详细信息。

```
aws ec2 describe-addresses \
    --filters "Name=domain,Values=vpc"
```

输出：

```
{
    "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"
        }
    ]
}
```

示例 3：检索由分配 ID 指定的弹性 IP 地址的详细信息

以下 describe-addresses 示例显示有关具有指定分配 ID 的弹性 IP 地址的详细信息，该地址与 EC2-VPC 中的实例相关联。

```
aws ec2 describe-addresses \
    --allocation-ids eipalloc-282d9641
```

输出：

```
{
    "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"
        }
    ]
}
```

示例 4：检索由 VPC 私有 IP 地址指定的弹性 IP 地址的详细信息

以下 describe-addresses 示例显示，与 EC2-VPC 中特定私有 IP 地址关联的弹性 IP 地址的详细信息。

```
aws ec2 describe-addresses \
    --filters "Name=private-ip-address,Values=10.251.50.12"
```

示例 5：检索 EC2-Classic 中有关弹性 IP 地址的详细信息

以下 describe-addresses 示例显示有关在 EC2-Classic 中使用的弹性 IP 地址的详细信息。

```
aws ec2 describe-addresses \
    --filters "Name=domain,Values=standard"
```

输出：

```
{
    "Addresses": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "PublicIp": "203.0.110.25",
            "PublicIpv4Pool": "amazon",
            "Domain": "standard"
        }
```

```
    ]
}
```

示例 6：检索由公有 IP 地址指定的弹性 IP 地址详细信息

以下 `describe-addresses` 示例显示有关具有值 `203.0.110.25` 的弹性 IP 地址的详细信息，该地址与 EC2-Classic 中的实例相关联。

```
aws ec2 describe-addresses \
    --public-ips 203.0.110.25
```

输出：

```
{
    "Addresses": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "PublicIp": "203.0.110.25",
            "PublicIpv4Pool": "amazon",
            "Domain": "standard"
        }
    ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeAddresses中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { DescribeAddressesCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";
```

```
export const main = async () => {
  const command = new DescribeAddressesCommand({
    // You can omit this property to show all addresses.
    AllocationIds: ["ALLOCATION_ID"],
  });

  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 (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DescribeAddresses中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了 EC2-Classic 中实例的指定弹性 IP 地址。

```
Get-EC2Address -AllocationId eipalloc-12345678
```

输出：

```
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
```

示例 2：此示例描述了 VPC 中实例的弹性 IP 地址。此语法需要 PowerShell 版本 3 或更高版本。

```
Get-EC2Address -Filter @{ Name="domain";Values="vpc" }
```

示例 3：此示例描述了 EC2-Classic 中实例的指定弹性 IP 地址。

```
Get-EC2Address -PublicIp 203.0.113.17
```

输出：

```
AllocationId            :
AssociationId           :
Domain                  : standard
InstanceId              : i-12345678
NetworkInterfaceId      :
NetworkInterfaceOwnerId :
PrivateIpAddress        :
PublicIp                : 203.0.113.17
```

示例 4：此示例描述了 EC2-Classic 中实例的弹性 IP 地址。此语法需要 PowerShell 版本 3 或更高版本。

```
Get-EC2Address -Filter @{ Name="domain";Values="standard" }
```

示例 5：此示例描述了您的所有弹性 IP 地址。

```
Get-EC2Address
```

示例 6：此示例返回过滤器中提供的实例 ID 的公有和私有 IP

```
Get-EC2Address -Region eu-west-1 -Filter @{Name="instance-
id";Values="i-0c12d3f4f567ffb89"} | Select-Object PrivateIpAddress, PublicIp
```

输出：

```
PrivateIpAddress PublicIp
---------------- --------
10.0.0.99        63.36.5.227
```

示例 7：此示例检索所有弹性 IP 及其分配 ID、关联 ID 和实例 ID

```
Get-EC2Address -Region eu-west-1 | Select-Object InstanceId, AssociationId,
 AllocationId, PublicIp
```

输出：

```
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
```

示例 8：此示例获取与标签键 “类别” 与值为 “Prod” 的标签密钥 “类别” 匹配的 EC2 IP 地址列表

```
Get-EC2Address -Filter @{Name="tag:Category";Values="Prod"}
```

输出：

```
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}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeAddresses中的。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用DescribeAddresses于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeAvailabilityZones`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeAvailabilityZones`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 构建和管理弹性服务

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    /// <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()
    {
        var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
            new DescribeAvailabilityZonesRequest());
        return zoneResponse.AvailabilityZones.Select(z => z.ZoneName).ToList();
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考DescribeAvailabilityZones中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
```

```
    Aws::EC2::Model::DescribeAvailabilityZonesRequest describe_request;
    auto describe_outcome =
ec2Client.DescribeAvailabilityZones(describe_request);

    if (describe_outcome.IsSuccess()) {
        std::cout << std::left <<
                  std::setw(32) << "ZoneName" <<
                  std::setw(20) << "State" <<
                  std::setw(32) << "Region" << std::endl;

        const auto &zones =
                describe_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:" <<
                  describe_outcome.GetError().GetMessage() << std::endl;
        result = false;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考[DescribeAvailabilityZones](https://docs.aws.amazon.com/goto/SdkForCpp/ec2-2016-11-15/DescribeAvailabilityZones)中的。

CLI

AWS CLI

描述可用区

以下示例 `describe-availability-zones` 显示了可供您使用的可用区详细信息。响应仅包括当前区域的可用区。在本示例中，将使用配置文件默认的 `us-west-2`（俄勒冈州）区域。

```
aws ec2 describe-availability-zones
```

输出：

```
{
    "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"
        }
    ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeAvailabilityZones中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了当前区域中可供您使用的可用区。

```
Get-EC2AvailabilityZone
```

输出：

```
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
```

示例 2：此示例描述了所有处于受损状态的可用区。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
Get-EC2AvailabilityZone -Filter @{ Name="state";Values="impaired" }
```

示例 3：在 PowerShell 版本 2 中，必须使用 New-Object 来创建过滤器。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = "impaired"

Get-EC2AvailabilityZone -Filter $filter
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DescribeAvailabilityZones](#)中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class AutoScaler:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :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
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"
        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"
        self.key_pair_name = f"{resource_prefix}-key-pair"


    def get_availability_zones(self):
        """
        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"]]
        except ClientError as err:
            raise AutoScalerError(f"Couldn't get availability zones: {err}.")
        else:
            return zones
```

- 有关 API 的详细信息，请参阅适用DescribeAvailabilityZones于 Python 的AWS SDK (Boto3) API 参考。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用DescribeAvailabilityZones于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeBundleTasks`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeBundleTasks`。

CLI

AWS CLI

描述您的捆绑包任务

此示例描述了您的所有捆绑包任务。

命令:

```
aws ec2 describe-bundle-tasks
```

输出：

```
{
  "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"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeBundleTasks中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的捆绑任务。

```
Get-EC2BundleTask -BundleId bun-12345678
```

示例 2：此示例描述了状态为 “完成” 或 “失败” 的捆绑任务。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "complete", "failed" )
```

```
Get-EC2BundleTask -Filter $filter
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeBundleTasks中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeCapacityReservations与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeCapacityReservations。

CLI

AWS CLI

示例 1：描述您的一个或多个容量预留

以下describe-capacity-reservations示例显示了有关您在当前 AWS 地区的所有容量预留的详细信息。

```
aws ec2 describe-capacity-reservations
```

输出：

```
{
    "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"
        }
    ]
}
```

示例 2：描述您的一个或多个容量预留

以下describe-capacity-reservations示例显示了有关指定容量预留的详细信息。

```
aws ec2 describe-capacity-reservations \
    --capacity-reservation-ids cr-1234abcd56EXAMPLE
```

输出：

```
{
    "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"
        }
    ]
}
```

有关更多信息，请参阅适用于 Linux 实例的 Amazon 弹性计算云用户指南中的查看容量预留。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeCapacityReservations中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了您为该地区预留的一个或多个容量

```
Get-EC2CapacityReservation -Region eu-west-1
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeCapacityReservations中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeCustomerGateways与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeCustomerGateways。

CLI

AWS CLI

描述您的客户网关

此示例描述了您的客户网关。

命令:

```
aws ec2 describe-customer-gateways
```

输出：

```
{
    "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"
        }
    ]
}
```

描述特定的客户网关

此示例描述了指定的客户网关。

命令:

```
aws ec2 describe-customer-gateways --customer-gateway-ids cgw-0e11f167
```

输出：

```
{
    "CustomerGateways": [
        {
            "CustomerGatewayId": "cgw-0e11f167",
            "IpAddress": "12.1.2.3",
            "State": "available",
            "Type": "ipsec.1",
            "BgpAsn": "65534"
        }
    ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeCustomerGateways中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的客户网关。

```
Get-EC2CustomerGateway -CustomerGatewayId cgw-1a2b3c4d
```

输出：

```
BgpAsn            : 65534
CustomerGatewayId : cgw-1a2b3c4d
IpAddress         : 203.0.113.12
State             : available
Tags              : {}
Type              : ipsec.1
```

示例 2：此示例描述了状态为 “待定” 或 “可用” 的所有客户网关。

```
$filter = New-Object Amazon.EC2.Model.Filter
```

```
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2CustomerGateway -Filter $filter
```

示例 3：此示例描述了您的所有客户网关。

```
Get-EC2CustomerGateway
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeCustomerGateways中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeDhcpOptions与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeDhcpOptions。

CLI

AWS CLI

示例 1：描述您的 DHCP 选项

以下describe-dhcp-options示例检索有关您的 DHCP 选项的详细信息。

```
aws ec2 describe-dhcp-options
```

输出：

```
{
    "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"
        }
    ]
}
```

有关更多信息，请参阅《AWS VPC 用户指南》中的 “使用 DHCP 选项集”。

示例 2：描述您的 DHCP 选项并筛选输出

以下describe-dhcp-options示例描述了您的 DHCP 选项，并使用筛选器仅返回example.com适用于域名服务器的 DHCP 选项。该示例使用--query参数在输出中仅显示配置信息和 ID。

```
aws ec2 describe-dhcp-options \
    --filters Name=key,Values=domain-name-servers Name=value,Values=example.com \
    --query "DhcpOptions[*].[DhcpConfigurations,DhcpOptionsId]"
```

输出：

```
[
    [
        [
            {
                "Key": "domain-name",
                "Values": [
                    {
                        "Value": "example.com"
                    }
                ]
            },
            {
                "Key": "domain-name-servers",
                "Values": [
                    {
                        "Value": "172.16.16.16"
                    }
                ]
            }
        ],
        "dopt-001122334455667ab"
    ]
]
```

有关更多信息，请参阅《AWS VPC 用户指南》中的 “使用 DHCP 选项集”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeDhcpOptions中的。

PowerShell

用于 PowerShell

示例 1：此示例列出了您的 DHCP 选项集。

```
Get-EC2DhcpOption
```

输出：

```
DhcpConfigurations                    DhcpOptionsId    Tag
------------------                    -------------    ---
{domain-name, domain-name-servers}    dopt-1a2b3c4d    {}
{domain-name, domain-name-servers}    dopt-2a3b4c5d    {}
{domain-name-servers}                 dopt-3a4b5c6d    {}
```

示例 2：此示例获取指定 DHCP 选项集的配置详细信息。

```
(Get-EC2DhcpOption -DhcpOptionsId dopt-1a2b3c4d).DhcpConfigurations
```

输出：

```
Key                  Values
---                  ------
domain-name          {abc.local}
domain-name-servers  {10.0.0.101, 10.0.0.102}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeDhcpOptions中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeFlowLogs`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeFlowLogs`。

CLI

AWS CLI

示例 1：描述您的所有流日志

以下describe-flow-logs示例显示了所有流日志的详细信息。

```
aws ec2 describe-flow-logs
```

输出：

```
{
    "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}"
        }
    ]
}
```

示例 2：描述您的流日志的子集

以下`describe-flow-logs`示例使用筛选器仅显示 Amazon CloudWatch Logs 中指定日志组中的流日志的详细信息。

```
aws ec2 describe-flow-logs \
    --filter "Name=log-group-name,Values=MyFlowLogs"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeFlowLogs](#)中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了一个或多个日志目标类型为 “s3” 的流日志

```
Get-EC2FlowLog -Filter @{Name="log-destination-type";Values="s3"}
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DescribeFlowLogs](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeHostReservationOfferings与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeHostReservationOfferings。

CLI

AWS CLI

描述专用主机预留服务

此示例描述了可供购买的 M4 实例系列的专用主机预留。

命令:

```
aws ec2 describe-host-reservation-offerings --filter Name=instance-
family,Values=m4
```

输出：

```
{
  "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
        }
    ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeHostReservationOfferings](#)中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了可为给定过滤器 “实例系列” 购买的专用主机预留 PaymentOption ，其中 “” NoUpfront

```
Get-EC2HostReservationOffering -Filter @{Name="instance-family";Values="m4"} |
 Where-Object PaymentOption -eq NoUpfront
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeHostReservationOfferings中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeHosts与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeHosts。

CLI

AWS CLI

查看有关专用主机的详细信息

以下describe-hosts示例显示您 AWS 账户中available专用主机的详细信息。

```
aws ec2 describe-hosts --filter "Name=state,Values=available"
```

输出：

```
{
    "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"
        }
    ]
}
```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon 弹性计算云用户指南》中的查看专用主机。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeHosts中的。

PowerShell

用于 PowerShell

示例 1：此示例返回 EC2 主机详细信息

```
Get-EC2Host
```

输出：

```
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              : {}
```

示例 2：此示例查询主机 h-01e23f4c AvailableInstanceCapacity d567899f1

```
Get-EC2Host -HostId h-01e23f4cd567899f1 | Select-Object -ExpandProperty
 AvailableCapacity | Select-Object -expand AvailableInstanceCapacity
```

输出：

```
AvailableCapacity InstanceType TotalCapacity
----------------- ------------ -------------
11                m4.xlarge    11
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeHosts中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeIamInstanceProfileAssociations与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeIamInstanceProfileAssociations。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 构建和管理弹性服务

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    /// <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)
    {
        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];
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考DescribeIamInstanceProfileAssociations中的。

CLI

AWS CLI

描述 IAM 实例配置文件关联

本示例描述了所有 IAM 实例配置文件关联。

命令:

```
aws ec2 describe-iam-instance-profile-associations
```

输出：

```
{
  "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"
          }
      }
   ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeIamInstanceProfileAssociations中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
const ec2Client = new EC2Client({});
const { IamInstanceProfileAssociations } = await ec2Client.send(
  new DescribeIamInstanceProfileAssociationsCommand({
    Filters: [
      { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
    ],
  }),
);
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DescribeIamInstanceProfileAssociations中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class AutoScaler:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
```

```
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :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
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"
        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"
        self.key_pair_name = f"{resource_prefix}-key-pair"


    def get_instance_profile(self, instance_id):
        """
        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]}]
            )
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't get instance profile association for instance
{instance_id}: {err}"
            )
        else:
            return response["IamInstanceProfileAssociations"][0]
```

- 有关 API 的详细信息，请参阅适用DescribeIamInstanceProfileAssociations于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeIdFormat`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeIdFormat`。

CLI

AWS CLI

示例 1：描述资源的 ID 格式

以下`describe-id-format`示例描述了安全组的 ID 格式。

```
aws ec2 describe-id-format \
    --resource security-group
```

在以下示例输出中，该`Deadline`值表示该资源类型从短 ID 格式永久切换到长 ID 格式的最后期限已于 2018 年 8 月 15 日 00:00 UTC 到期。

```
{
    "Statuses": [
```

```
        {
            "Deadline": "2018-08-15T00:00:00.000Z",
            "Resource": "security-group",
            "UseLongIds": true
        }
    ]
}
```

示例 2：描述所有资源的 ID 格式

以下`describe-id-format`示例描述了所有资源类型的 ID 格式。所有支持短 ID 格式的资源类型均已切换为使用长 ID 格式。

```
aws ec2 describe-id-format
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeIdFormat](#)中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定资源类型的 ID 格式。

```
Get-EC2IdFormat -Resource instance
```

输出：

```
Resource       UseLongIds
--------       ----------
instance       False
```

示例 2：此示例描述了所有支持加长 ID 的资源类型的 ID 格式。

```
Get-EC2IdFormat
```

输出：

```
Resource       UseLongIds
--------       ----------
reservation    False
```

```
instance        False
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeIdFormat中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeIdentityIdFormat`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeIdentityIdFormat`。

CLI

AWS CLI

描述 IAM 角色的 ID 格式

以下`describe-identity-id-format`示例描述了您的 AWS 账户中的 IAM 角色`EC2Role`创建的实例所接收的 ID 格式。

```
aws ec2 describe-identity-id-format \
    --principal-arn arn:aws:iam::123456789012:role/my-iam-role \
    --resource instance
```

以下输出表明由此角色创建的实例会收到长 ID 格式的 ID。

```
{
    "Statuses": [
        {
            "Deadline": "2016-12-15T00:00:00Z",
            "Resource": "instance",
            "UseLongIds": true
        }
    ]
}
```

描述 IAM 用户的 ID 格式

以下`describe-identity-id-format`示例描述了 IAM 用户`AdminUser`在您的 AWS 账户中创建的快照所接收的 ID 格式。

```
aws ec2 describe-identity-id-format \
    --principal-arn arn:aws:iam::123456789012:user/AdminUser \
    --resource snapshot
```

输出表明该用户创建的快照会收到长 ID 格式的 ID。

```
{
    "Statuses": [
        {
            "Deadline": "2016-12-15T00:00:00Z",
            "Resource": "snapshot",
            "UseLongIds": true
        }
    ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeIdentityIdFormat中的。

PowerShell

用于 PowerShell

示例 1：此示例返回给定角色的资源 “图片” 的 ID 格式

```
Get-EC2IdentityIdFormat -PrincipalArn arn:aws:iam::123456789511:role/JDBC -
Resource image
```

输出：

```
Deadline             Resource UseLongIds
--------             -------- ----------
8/2/2018 11:30:00 PM image    True
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeIdentityIdFormat中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeImageAttribute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeImageAttribute`。

CLI

AWS CLI

描述 AMI 的启动权限

此示例描述了指定 AMI 的启动权限。

命令:

```
aws ec2 describe-image-attribute --image-id ami-5731123e --attribute
 launchPermission
```

输出：

```
{
    "LaunchPermissions": [
        {
            "UserId": "123456789012"
        }
    ],
    "ImageId": "ami-5731123e",
}
```

描述 AMI 的产品代码

此示例描述了指定 AMI 的产品代码。请注意，此 AMI 没有产品代码。

命令:

```
aws ec2 describe-image-attribute --image-id ami-5731123e --attribute productCodes
```

输出：

```
{
    "ProductCodes": [],
    "ImageId": "ami-5731123e",
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeImageAttribute中的。

PowerShell

用于 PowerShell

示例 1：此示例获取指定 AMI 的描述。

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute description
```

输出：

```
BlockDeviceMappings : {}
Description         : My image description
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

示例 2：此示例获取指定 AMI 的启动权限。

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

输出：

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {all}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     :
```

示例 3：此示例测试是否启用了增强联网。

```
Get-EC2ImageAttribute -ImageId ami-12345678 -Attribute sriovNetSupport
```

输出：

```
BlockDeviceMappings : {}
Description         :
ImageId             : ami-12345678
KernelId            :
LaunchPermissions   : {}
ProductCodes        : {}
RamdiskId           :
SriovNetSupport     : simple
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeImageAttribute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeImages`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeImages`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用实例

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
#############################################################################
# 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 getopts 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
}
```

本示例中使用的实用程序函数。

```
##############################################################################
# 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
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeImages中的。

CLI

AWS CLI

示例 1：描述 AMI

以下 `describe-images` 示例描述了指定区域中的指定 AMI。

```
aws ec2 describe-images \
    --region us-east-1 \
    --image-ids ami-1234567890EXAMPLE
```

输出：

```
{
    "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-1112223334444aaabb",
                        "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"
        }
    ]
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的亚马逊机器映像（AMI）。

示例 2：根据筛选器描述 AMI

以下 `describe-images` 示例描述了由 Amazon 提供，并受 Amazon EBS 支持的 Windows AMI。

```
aws ec2 describe-images \
    --owners amazon \
    --filters "Name=platform,Values=windows" "Name=root-device-type,Values=ebs"
```

有关 `describe-images` 的输出示例，请参阅示例 1。

有关使用筛选器的其他示例，请参阅《Amazon EC2 用户指南》中的列出和筛选资源。

示例 3：根据标签描述 AMI

以下 `describe-images` 示例描述了带有标签 `Type=Custom` 的所有 AMI。示例使用 `--query` 参数仅显示 AMI ID。

```
aws ec2 describe-images \
    --filters "Name=tag:Type,Values=Custom" \
    --query 'Images[*].[ImageId]' \
    --output text
```

输出：

```
ami-1234567890EXAMPLE
ami-0abcdef1234567890
```

有关使用标签筛选器的更多示例，请参阅《Amazon EC2 用户指南》中的使用标签。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeImages中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { paginateDescribeImages } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

// List at least the first i386 image available for EC2 instances.
export const main = async () => {
  // 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: 25 },
    {
      // 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: ["x86_64"] }],
    },
  );

  try {
    const arm64Images = [];
    for await (const page of paginator) {
      if (page.Images.length) {
        arm64Images.push(...page.Images);
        // Once we have at least 1 result, we can stop.
        if (arm64Images.length >= 1) {
          break;
        }
      }
    }
    console.log(arm64Images);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DescribeImages中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的 AMI。

```
Get-EC2Image -ImageId ami-12345678
```

输出：

```
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
```

示例 2：此示例描述了您拥有的 AMI。

```
Get-EC2Image -owner self
```

示例 3：此示例描述了运行微软 Windows 服务器的公共 AMI。

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

示例 4：此示例描述了 “us-west-2” 区域中的所有公有 AMI。

```
Get-EC2Image -Region us-west-2
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeImages中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance
 actions."""

    def __init__(self, ec2_resource, instance=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
 resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param instance: A Boto3 Instance object. This is a high-level object
 that
                          wraps instance actions.
        """
        self.ec2_resource = ec2_resource
        self.instance = instance

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource)


    def get_images(self, image_ids):
        """
        Gets information about Amazon Machine Images (AMIs) from a list of AMI
 IDs.

        :param image_ids: The list of AMIs to look up.
        :return: A list of Boto3 Image objects that represent the requested AMIs.
        """
        try:
```

```
            images = list(self.ec2_resource.images.filter(ImageIds=image_ids))
        except ClientError as err:
            logger.error(
                "Couldn't get images. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return images
```

- 有关 API 的详细信息，请参阅适用DescribeImages于 Python 的AWS SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
// A simple Rust program to list all Amazon images in the currently configured
 region.
#[tokio::main]
async fn main() -> Result<(), Error> {
    let config = aws_config::load_from_env().await;
    let client = Client::new(&config);

    let resp = client.describe_images().owners("amazon").send().await?;

    println!("AWS SDK for Rust v{}", PKG_VERSION);
    println!("Describing Amazon Machine Images (AMIs):");

    let mut images: Vec<_> = resp
        .images()
        .iter()
```

```
        .filter(|i| {
            i.description()
                .filter(|i| i.contains("Amazon Linux AMI 2023"))
                .is_some()
        })
        .collect();
    images.sort_by(|a, b| a.description.cmp(&b.description));

    if images.is_empty() {
        println!("No images found.");
        return Ok(());
    }

    for image in images {
        let id = image.image_id().unwrap_or_default();
        let description = image.description().unwrap_or_default();

        println!("{id}: {description}");
    }

    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用DescribeImages于 Rust 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeImportImageTasks`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeImportImageTasks`。

CLI

AWS CLI

监视导入图像任务

以下`describe-import-image-tasks`示例检查指定导入映像任务的状态。

```
aws ec2 describe-import-image-tasks \
```

```
    --import-task-ids import-ami-1234567890abcdef0
```

正在执行的导入图像任务的输出。

```
{
    "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"
        }
    ]
}
```

已完成的导入图像任务的输出。生成的 AMI 的 ID 由提供ImageId。

```
{
    "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"
        }
    ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeImportImageTasks](#)中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的图像导入任务。

```
Get-EC2ImportImageTask -ImportTaskId import-ami-hgfedcba
```

输出：

```
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   :
```

示例 2：此示例描述了您的所有图像导入任务。

```
Get-EC2ImportImageTask
```

输出：

```
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   :
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeImportImageTasks中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeImportSnapshotTasks`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeImportSnapshotTasks`。

CLI

AWS CLI

监视导入快照任务

以下`describe-import-snapshot-tasks`示例检查指定导入快照任务的状态。

```
aws ec2 describe-import-snapshot-tasks \
    --import-task-ids import-snap-1234567890abcdef0
```

正在进行的导入快照任务的输出：

```
{
    "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/converting",
                "UserBucket": {
                    "S3Bucket": "my-import-bucket",
                    "S3Key": "vms/my-server-vm.vmdk"
                }
            }
        }
    ]
}
```

已完成的导入快照任务的输出。生成的快照的 ID 由提供SnapshotId。

```
{
    "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"
                }
            }
        }
    ]
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeImportSnapshotTasks](#)中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的快照导入任务。

```
Get-EC2ImportSnapshotTask -ImportTaskId import-snap-abcdefgh
```

输出：

```
Description            ImportTaskId              SnapshotTaskDetail

-----------------      --------------------      ------------------

Disk Image Import 1    import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

示例 2：此示例描述了您的所有快照导入任务。

```
Get-EC2ImportSnapshotTask
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DescribeImportSnapshotTasks](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeInstanceAttribute与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeInstanceAttribute。

CLI

AWS CLI

描述实例类型

此示例描述了指定实例的实例类型。

命令:

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --attribute
 instanceType
```

输出：

```
{
    "InstanceId": "i-1234567890abcdef0"
    "InstanceType": {
        "Value": "t1.micro"
    }
}
```

描述该 disableApiTermination 属性

此示例描述了指定实例的disableApiTermination属性。

命令:

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --attribute
 disableApiTermination
```

输出：

```
{
```

```
"InstanceId": "i-1234567890abcdef0"
    "DisableApiTermination": {
        "Value": "false"
    }
}
```

描述实例的块储存设备映射

此示例描述了指定实例的blockDeviceMapping属性。

命令:

```
aws ec2 describe-instance-attribute --instance-id i-1234567890abcdef0 --attribute
 blockDeviceMapping
```

输出：

```
{
    "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"
            }
        }
    ],
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeInstanceAttribute中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定实例的实例类型。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

输出：

```
InstanceType                     : t2.micro
```

示例 2：此示例描述了是否为指定实例启用了增强联网。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

输出：

```
SriovNetSupport                  : simple
```

示例 3：此示例描述了指定实例的安全组。

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

输出：

```
GroupId
-------
sg-12345678
sg-45678901
```

示例 4：此示例描述了是否为指定实例启用 EBS 优化。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

输出：

```
EbsOptimized                     : False
```

示例 5：此示例描述了指定实例disableApiTermination的 “” 属性。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute disableApiTermination
```

输出：

```
DisableApiTermination            : False
```

示例 6：此示例描述了指定实例instanceInitiatedShutdown的 “行为” 属性。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

输出：

```
InstanceInitiatedShutdownBehavior : stop
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeInstanceAttribute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeInstanceStatus`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeInstanceStatus`。

CLI

AWS CLI

描述实例的状态

以下 `describe-instance-status` 示例描述了指定实例的当前状态。

```
aws ec2 describe-instance-status \
    --instance-ids i-1234567890abcdef0
```

输出：

```
{
    "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"
                    }
                ]
            }
        }
    ]
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的监控实例状态。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeInstanceStatus中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定实例的状态。

```
Get-EC2InstanceStatus -InstanceId i-12345678
```

输出：

```
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
```

输出：

```
Code    Name
----    ----
16      running
```

```
$status.Status
```

输出：

```
Details           Status
-------           ------
{reachability}    ok
```

```
$status.SystemStatus
```

输出：

```
Details           Status
-------           ------
{reachability}    ok
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DescribeInstanceStatus](#)中的。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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(())
```

```
}
```

- 有关 API 的详细信息，请参阅适用DescribeInstanceStatus于 Rust 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeInstanceTypes与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeInstanceTypes。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用实例

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    /// <summary>
    /// Describe the instance types available.
    /// </summary>
    /// <returns>A list of instance type information.</returns>
    public async Task<List<InstanceTypeInfo>>
DescribeInstanceTypes(ArchitectureValues architecture)
    {
        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;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考DescribeInstanceTypes中的。

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
###############################################################################
# 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
}
```

本示例中使用的实用程序函数。

```
################################################################################
# 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
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeInstanceTypes](#)中的。

CLI

AWS CLI

示例 1：描述实例类型

以下 `describe-instance-types` 示例显示指定实例类型的详细信息。

```
aws ec2 describe-instance-types \
    --instance-types t2.micro
```

输出：

```
{
    "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
        }
```

```
    ]
}
```

有关更多信息，请参阅适用于 Linux 实例的 Amazon 弹性计算云用户指南中的实例类型。

示例 2：筛选可用的实例类型

可以指定筛选器，将结果范围限定为具有特定特征的实例类型。以下 describe-instance-types 示例列出支持休眠的实例类型。

```
aws ec2 describe-instance-types \
    --filters Name=hibernation-supported,Values=true --query
 'InstanceTypes[*].InstanceType'
```

输出：

```
[
    "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"
]
```

有关更多信息，请参阅适用于 Linux 实例的 Amazon 弹性计算云用户指南中的实例类型。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeInstanceTypes中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
   // Get a list of instance types.
   public static String getInstanceTypes(Ec2Client ec2) {
       String instanceType;
       try {
           DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
               .maxResults(10)
               .build();

           DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
           List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
           for (InstanceTypeInfo type : instanceTypes) {
               System.out.println("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
               System.out.println("Network information is " +
type.networkInfo().toString());
               System.out.println("Instance type is " +
type.instanceType().toString());
               instanceType = type.instanceType().toString();
               if (instanceType.compareTo("t2.2xlarge")  == 0){
                   return instanceType;
               }
           }

       } catch (SsmException e) {
           System.err.println(e.getMessage());
           System.exit(1);
       }
       return "";
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeInstanceTypes中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import {
  paginateDescribeInstanceTypes,
  DescribeInstanceTypesCommand,
} from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

// List at least the first arm64 EC2 instance type available.
export const main = async () => {
  // 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: 25 },
    {
      Filters: [
        { Name: "processor-info.supported-architecture", Values: ["x86_64"] },
        { Name: "free-tier-eligible", Values: ["true"] },
      ],
    }
  );

  try {
    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(instanceTypes);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DescribeInstanceTypes中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
// 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
    }
}
```

- 有关 API 的详细信息，请参阅适用DescribeInstanceTypes于 K otlin 的AWS SDK API 参考。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance
 actions."""

    def __init__(self, ec2_resource, instance=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
 resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param instance: A Boto3 Instance object. This is a high-level object
 that
                         wraps instance actions.
```

```
        """
        self.ec2_resource = ec2_resource
        self.instance = instance

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource)


    def get_instance_types(self, architecture):
        """
        Gets instance types that support the specified architecture and are
designated
        as either 'micro' or 'small'. When an instance is created, the instance
type
        you specify must support the architecture of the AMI you use.

        :param architecture: The kind of architecture the instance types must
support,
                             such as 'x86_64'.
        :return: A list of instance types that support the specified architecture
                 and are either 'micro' or 'small'.
        """
        try:
            inst_types = []
            it_paginator = self.ec2_resource.meta.client.get_paginator(
                "describe_instance_types"
            )
            for page in it_paginator.paginate(
                Filters=[
                    {
                        "Name": "processor-info.supported-architecture",
                        "Values": [architecture],
                    },
                    {"Name": "instance-type", "Values": ["*.micro", "*.small"]},
                ]
            ):
                inst_types += page["InstanceTypes"]
        except ClientError as err:
            logger.error(
                "Couldn't get instance types. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
        else:
            return inst_types
```

- 有关 API 的详细信息，请参阅适用DescribeInstanceTypes于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeInstances`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeInstances`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 构建和管理弹性服务
- 开始使用实例

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
/// <summary>
/// Get information about existing EC2 images.
/// </summary>
/// <returns>Async task.</returns>
public async Task DescribeInstances()
{
    // List all EC2 instances.
```

```
        await GetInstanceDescriptions();

        string tagName = "IncludeInList";
        string tagValue = "Yes";
        await GetInstanceDescriptionsFiltered(tagName, tagValue);
    }

    /// <summary>
    /// Get information for all existing Amazon EC2 instances.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task GetInstanceDescriptions()
    {
        Console.WriteLine("Showing all instances:");
        var paginator = _amazonEC2.Paginators.DescribeInstances(new
DescribeInstancesRequest());

        await foreach (var response in paginator.Responses)
        {
            foreach (var reservation in response.Reservations)
            {
                foreach (var instance in reservation.Instances)
                {
                    Console.Write($"Instance ID: {instance.InstanceId}");
                    Console.WriteLine($"\tCurrent State: {instance.State.Name}");
                }
            }
        }
    }

    /// <summary>
    /// Get information about EC2 instances filtered by a tag name and value.
    /// </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>Async task.</returns>
    public async Task GetInstanceDescriptionsFiltered(string tagName, string
tagValue)
    {
        // This tag filters the results of the instance list.
        var filters = new List<Filter>
        {
            new Filter
            {
```

```
                Name = $"tag:{tagName}",
                Values = new List<string>
                {
                    tagValue,
                },
            },
        };
        var request = new DescribeInstancesRequest
        {
            Filters = filters,
        };

        Console.WriteLine("\nShowing instances with tag: \"IncludeInList\" set to
\"Yes\".");
        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.Write($"Instance ID: {instance.InstanceId} ");
                    Console.WriteLine($"\tCurrent State: {instance.State.Name}");
                }
            }
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考DescribeInstances中的。

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
##############################################################################
# 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 getopts 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
}
```

本示例中使用的实用程序函数。

```
########################################################################################
# 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
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeInstances](#)中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeInstancesRequest request;
    bool header = false;
    bool done = false;
    while (!done) {
        auto 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;
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考DescribeInstances中的。

CLI

AWS CLI

示例 1：描述实例

以下 describe-instances 示例描述了指定的实例。

```
aws ec2 describe-instances \
    --instance-ids i-1234567890abcdef0
```

输出：

```
{
    "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-04a636d18e83cfacb",
                    "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"
        }
```

```
    ]
}
```

示例 2：筛选具有指定类型的实例

以下 describe-instances 示例使用筛选器，将结果范围限定为指定类型的实例。

```
aws ec2 describe-instances \
    --filters Name=instance-type,Values=m5.large
```

有关示例输出，请参阅示例 1。

有关更多信息，请参阅《Amazon EC2 用户指南》中的使用 CLI 列出和筛选。

示例 3：筛选具有指定类型和可用区的实例

以下 describe-instances 示例使用多个筛选器，将结果范围限定为同样位于指定可用区且具有指定类型的实例。

```
aws ec2 describe-instances \
    --filters Name=instance-type,Values=t2.micro,t3.micro Name=availability-
zone,Values=us-east-2c
```

有关示例输出，请参阅示例 1。

示例 4：使用 JSON 文件筛选具有指定类型和可用区的实例

以下 describe-instances 示例使用 JSON 输入文件执行与上一个示例相同的筛选。当筛选器变得更加复杂时，可以更容易地在 JSON 文件中加以指定。

```
aws ec2 describe-instances \
    --filters file://filters.json
```

filters.json 的内容：

```
[
    {
        "Name": "instance-type",
        "Values": ["t2.micro", "t3.micro"]
```

```
    },
    {
        "Name": "availability-zone",
        "Values": ["us-east-2c"]
    }
]
```

有关示例输出，请参阅示例 1。

示例 5：筛选具有指定 Owner 标签的实例

以下 describe-instances 示例使用标签筛选器，将结果范围限定为具有指定标签键（Owner）标签的实例，无论标签值如何。

```
aws ec2 describe-instances \
    --filters "Name=tag-key,Values=Owner"
```

有关示例输出，请参阅示例 1。

示例 6：筛选具有指定 my-team 标签值的实例

以下 describe-instances 示例使用标签筛选器，将结果范围限定为具有指定标签值（my-team）标签的实例，无论标签键如何。

```
aws ec2 describe-instances \
    --filters "Name=tag-value,Values=my-team"
```

有关示例输出，请参阅示例 1。

示例 7：筛选具有指定 Owner 标签和 my-team 值的实例

以下 describe-instances 示例使用标签筛选器，将结果范围限定为具有指定标签值（Owner=my-team）的实例。

```
aws ec2 describe-instances \
    --filters "Name=tag:Owner,Values=my-team"
```

有关示例输出，请参阅示例 1。

示例 8：仅显示所有实例的实例和子网 ID

以下 describe-instances 示例使用 --query 参数，以 JSON 格式仅显示所有实例的实例和子网 ID。

Linux 和 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
```

输出：

```
[
    {
        "Instance": "i-057750d42936e468a",
        "Subnet": "subnet-069beee9b12030077"
    },
    {
        "Instance": "i-001efd250faaa6ffa",
        "Subnet": "subnet-0b715c6b7db68927a"
    },
    {
        "Instance": "i-027552a73f021f3bd",
        "Subnet": "subnet-0250c25a1f4e15235"
    }
    ...
]
```

示例 9：筛选指定类型的实例并仅显示其实例 ID

以下 describe-instances 示例使用筛选器，将结果范围限定为指定类型的实例，并使用 --query 参数仅显示实例 ID。

```
aws ec2 describe-instances \
```

```
    --filters "Name=instance-type,Values=t2.micro" \
    --query "Reservations[*].Instances[*].[InstanceId]" \
    --output text
```

输出：

```
i-031c0dc19de2fb70c
i-00d8bff789a736b75
i-0b715c6b7db68927a
i-0626d4edd54f1286d
i-00b8ae04f9f99908e
i-0fc71c25d2374130c
```

示例 10：筛选指定类型的实例，并仅显示其实例 ID、可用区和指定的标签值

以下 `describe-instances` 示例以表格格式显示实例的实例 ID、可用区和 `Name` 标签值，这些实例有一个名为 `tag-key` 的标签。

Linux 和 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
```

输出：

```
------------------------------------------------------------
|                    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     |
+--------------+-----------------------+--------------------+
```

示例 11：描述分区置放群组中的实例

以下 describe-instances 示例描述了指定的实例。输出包括实例的置放信息，其中包含实例的置放群组名称和分区编号。

```
aws ec2 describe-instances \
    --instance-ids i-0123a456700123456 \
    --query "Reservations[*].Instances[*].Placement"
```

输出：

```
[
    [
        {
            "AvailabilityZone": "us-east-1c",
            "GroupName": "HDFS-Group-A",
            "PartitionNumber": 3,
            "Tenancy": "default"
        }

    ]
]
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的描述置放群组中的实例。

示例 12：筛选具有指定置放群组和分区编号的实例

以下 describe-instances 示例将结果筛选为仅具有指定置放群组和分区编号的实例。

```
aws ec2 describe-instances \
    --filters "Name=placement-group-name,Values=HDFS-Group-A" "Name=placement-
partition-number,Values=7"
```

以下仅显示输出中的相关信息。

```
"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"
        }
    ],
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的描述置放群组中的实例。

示例 13：筛选配置为允许从实例元数据访问标签的实例

以下 `describe-instances` 示例将结果筛选为，仅显示配置为允许从实例元数据访问实例标签的实例。

```
aws ec2 describe-instances \
    --filters "Name=metadata-options.instance-metadata-tags,Values=enabled" \
    --query "Reservations[*].Instances[*].InstanceId" \
    --output text
```

预期的输出如下所示。

```
i-1234567890abcdefg
i-abcdefg1234567890
i-11111111aaaaaaaaa
i-aaaaaaaa111111111
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的使用实例元数据中的实例标签。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeInstances中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
import software.amazon.awssdk.services.ec2.paginators.DescribeInstancesIterable;

/**
 * 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 DescribeInstances {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        describeEC2Instances(ec2);
        ec2.close();
    }

    public static void describeEC2Instances(Ec2Client ec2) {
        try {
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                .maxResults(10)
```

```
                .build();

            DescribeInstancesIterable instancesIterable =
 ec2.describeInstancesPaginator(request);
            instancesIterable.stream()
                .flatMap(r -> r.reservations().stream())
                .flatMap(reservation -> reservation.instances().stream())
                .forEach(instance -> {
                    System.out.println("Instance Id is " +
 instance.instanceId());
                    System.out.println("Image id is " + instance.imageId());
                    System.out.println("Instance type is " +
 instance.instanceType());
                    System.out.println("Instance state name is " +
 instance.state().name());
                    System.out.println("Monitoring information is " +
 instance.monitoring().state());
                });

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorCode());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeInstances中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { DescribeInstancesCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";
```

```
// List all of your EC2 instances running with x86_64 architecture that were
// launched this month.
export const main = async () => {
  const d = new Date();
  const year = d.getFullYear();
  const month = `0${d.getMonth() + 1}`.slice(-2);
  const launchTimePattern = `${year}-${month}-*`;
  const command = new DescribeInstancesCommand({
    Filters: [
      { Name: "architecture", Values: ["x86_64"] },
      { Name: "instance-state-name", Values: ["running"] },
      {
        Name: "launch-time",
        Values: [launchTimePattern],
      },
    ],
  });

  try {
    const { Reservations } = await client.send(command);
    const instanceList = Reservations.reduce((prev, current) => {
      return prev.concat(current.Instances);
    }, []);

    console.log(instanceList);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DescribeInstances中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅适用DescribeInstances于 K otlin 的AWS SDK API 参考。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的实例。

```
(Get-EC2Instance -InstanceId i-12345678).Instances
```

输出：

```
AmiLaunchIndex        : 0
Architecture          : x86_64
BlockDeviceMappings   : {/dev/sda1}
ClientToken           : TleEy1448154045270
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
```

示例 2：此示例描述了您在当前地区的所有实例，按预留量分组。要查看实例详细信息，请在每个预留对象中展开实例集合。

```
Get-EC2Instance
```

输出：

```
GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 226008221399
ReservationId : r-c5df370c
```

```
GroupNames    : {}
Groups        : {}
Instances     : {}
OwnerId       : 123456789012
RequesterId   : 854251627541
ReservationId : r-63e65bab
...
```

示例 3：此示例说明如何使用筛选器查询 VPC 的特定子网中的 EC2 实例。

```
(Get-EC2Instance -Filter @{Name="vpc-id";Values="vpc-1a2bc34d"},@{Name="subnet-
id";Values="subnet-1a2b3c4d"}).Instances
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DescribeInstances](#)中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class InstanceWrapper:
```

```
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance
actions."""

    def __init__(self, ec2_resource, instance=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param instance: A Boto3 Instance object. This is a high-level object
that
                            wraps instance actions.
        """
        self.ec2_resource = ec2_resource
        self.instance = instance

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource)


    def display(self, indent=1):
        """
        Displays information about an instance.

        :param indent: The visual indent to apply to the output.
        """
        if self.instance is None:
            logger.info("No instance to display.")
            return

        try:
            self.instance.load()
            ind = "\t" * indent
            print(f"{ind}ID: {self.instance.id}")
            print(f"{ind}Image ID: {self.instance.image_id}")
            print(f"{ind}Instance type: {self.instance.instance_type}")
            print(f"{ind}Key name: {self.instance.key_name}")
            print(f"{ind}VPC ID: {self.instance.vpc_id}")
            print(f"{ind}Public IP: {self.instance.public_ip_address}")
            print(f"{ind}State: {self.instance.state['Name']}")
        except ClientError as err:
            logger.error(
```

```
                "Couldn't display your instance. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- 有关 API 的详细信息，请参阅适用DescribeInstances于 Python 的AWS SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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
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__
```

- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考DescribeInstances中的。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
async fn show_state(client: &Client, ids: Option<Vec<String>>) -> Result<(),
 Error> {
    let resp = client
        .describe_instances()
        .set_instance_ids(ids)
        .send()
```

```
        .await?;

    for reservation in resp.reservations() {
        for instance in reservation.instances() {
            println!("Instance ID: {}", instance.instance_id().unwrap());
            println!(
                "State:       {:?}",
                instance.state().unwrap().name().unwrap()
            );
            println!();
        }
    }

    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用DescribeInstances于 Rust 的AWS SDK API 参考。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    TRY.
        oo_result = lo_ec2->describeinstances( ) .                      "
oo_result is returned for testing purposes. "

        " Retrieving details of EC2 instances. "
        DATA: lv_istance_id    TYPE /aws1/ec2string,
              lv_status        TYPE /aws1/ec2instancestatename,
              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_istance_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( ).
          ENDLOOP.
        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.
```

- 有关 API 的详细信息，请参阅适用DescribeInstances于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeInternetGateways与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeInternetGateways。

CLI

AWS CLI

描述互联网网关

以下describe-internet-gateways示例描述了指定的互联网网关。

```
aws ec2 describe-internet-gateways \
    --internet-gateway-ids igw-0d0fb496b3EXAMPLE
```

输出：

```
{
    "InternetGateways": [
        {
            "Attachments": [
                {
                    "State": "available",
                    "VpcId": "vpc-0a60eb65b4EXAMPLE"
                }
```

```
            ],
            "InternetGatewayId": "igw-0d0fb496b3EXAMPLE",
            "OwnerId": "123456789012",
            "Tags": [
                {
                    "Key": "Name",
                    "Value": "my-igw"
                }
            ]
        }
    ]
}
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的互联网网关。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeInternetGateways中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的互联网网关。

```
Get-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d
```

输出：

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
```

示例 2：此示例描述了您的所有互联网网关。

```
Get-EC2InternetGateway
```

输出：

```
Attachments       InternetGatewayId     Tags
-----------       -----------------     ----
{vpc-1a2b3c4d}    igw-1a2b3c4d          {}
{}                igw-2a3b4c5d          {}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeInternetGateways中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeKeyPairs`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeKeyPairs`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用实例

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
/// <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)
{
    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();
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考DescribeKeyPairs中的。

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
###############################################################################
# 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 getopts 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 getopts "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
}
```

本示例中使用的实用程序函数。

```
###############################################################################
# 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
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeKeyPairs中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DescribeKeyPairsRequest request;

auto 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;
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考DescribeKeyPairs中的。

CLI

AWS CLI

显示密钥对

以下 `describe-key-pairs` 示例显示有关指定密钥对的信息。

```
aws ec2 describe-key-pairs \
    --key-names my-key-pair
```

输出：

```
{
    "KeyPairs": [
        {
            "KeyPairId": "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"
        }
    ]
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的描述公有密钥。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeKeyPairs中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    public static void describeKeys(Ec2Client ec2) {
        try {
            DescribeKeyPairsResponse response = ec2.describeKeyPairs();
            response.keyPairs().forEach(keyPair -> System.out.printf(
                    "Found key pair with name %s " +
                            "and fingerprint %s",
                    keyPair.keyName(),
                    keyPair.keyFingerprint()));
```

```
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeKeyPairs中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { DescribeKeyPairsCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

export const main = async () => {
  const command = new DescribeKeyPairsCommand({});

  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 (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DescribeKeyPairs中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅适用DescribeKeyPairs于 K otlin 的AWS SDK API 参考。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的 key pair。

```
Get-EC2KeyPair -KeyName my-key-pair
```

输出：

```
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
```

示例 2：此示例描述了您的所有密钥对。

```
Get-EC2KeyPair
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeKeyPairs中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class KeyPairWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair
 actions."""

    def __init__(self, ec2_resource, key_file_dir, key_pair=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
 resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param key_file_dir: The folder where the private key information is
 stored.
                             This should be a secure folder.
        :param key_pair: A Boto3 KeyPair object. This is a high-level object that
                         wraps key pair actions.
        """
        self.ec2_resource = ec2_resource
        self.key_pair = key_pair
        self.key_file_path = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource, tempfile.TemporaryDirectory())


    def list(self, limit):
```

```
        """
        Displays a list of key pairs for the current account.

        :param limit: The maximum number of key pairs to list.
        """
        try:
            for kp in self.ec2_resource.key_pairs.limit(limit):
                print(f"Found {kp.key_type} key {kp.name} with fingerprint:")
                print(f"\t{kp.key_fingerprint}")
        except ClientError as err:
            logger.error(
                "Couldn't list key pairs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- 有关 API 的详细信息，请参阅适用DescribeKeyPairs于 Python 的AWS SDK (Boto3) API 参考。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用DescribeKeyPairs于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeNetworkAcls与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeNetworkAcls。

CLI

AWS CLI

描述您的网络 ACL

以下describe-network-acls示例检索有关您的网络 ACL 的详细信息。

```
aws ec2 describe-network-acls
```

输出：

```
{
    "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"
        }
    ]
}
```

有关更多信息，请参阅《AWS VPC 用户指南》中的网络 ACL。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeNetworkAcls中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的网络 ACL。

```
Get-EC2NetworkAcl -NetworkAclId acl-12345678
```

输出：

```
Associations : {aclassoc-1a2b3c4d}
Entries      : {Amazon.EC2.Model.NetworkAclEntry,
 Amazon.EC2.Model.NetworkAclEntry}
IsDefault    : False
NetworkAclId : acl-12345678
Tags         : {Name}
VpcId        : vpc-12345678
```

示例 2：此示例描述了指定网络 ACL 的规则。

```
(Get-EC2NetworkAcl -NetworkAclId acl-12345678).Entries
```

输出：

```
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
```

示例 3：此示例描述了您的所有网络 ACL。

```
Get-EC2NetworkAcl
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DescribeNetworkAcls](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeNetworkInterfaceAttribute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeNetworkInterfaceAttribute`。

CLI

AWS CLI

描述网络接口的连接属性

此示例命令描述了指定网络接口的`attachment`属性。

命令:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
 --attribute attachment
```

输出：

```
{
  "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"
  }
}
```

描述网络接口的描述属性

此示例命令描述了指定网络接口的description属性。

命令:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
 --attribute description
```

输出：

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "Description": {
      "Value": "My description"
  }
}
```

描述网络接口的 GroupSet 属性

此示例命令描述了指定网络接口的groupSet属性。

命令:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
 --attribute groupSet
```

输出：

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "Groups": [
```

```
        {
            "GroupName": "my-security-group",
            "GroupId": "sg-903004f8"
        }
    ]
}
```

描述网络接口的 sourceDestCheck 属性

此示例命令描述了指定网络接口的sourceDestCheck属性。

命令:

```
aws ec2 describe-network-interface-attribute --network-interface-id eni-686ea200
 --attribute sourceDestCheck
```

输出：

```
{
  "NetworkInterfaceId": "eni-686ea200",
  "SourceDestCheck": {
      "Value": true
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeNetworkInterfaceAttribute中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的网络接口。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Attachment
```

输出：

```
Attachment         : Amazon.EC2.Model.NetworkInterfaceAttachment
```

示例 2：此示例描述了指定的网络接口。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 Description
```

输出：

```
Description        : My description
```

示例 3：此示例描述了指定的网络接口。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 GroupSet
```

输出：

```
Groups             : {my-security-group}
```

示例 4：此示例描述了指定的网络接口。

```
Get-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-12345678 -Attribute
 SourceDestCheck
```

输出：

```
SourceDestCheck    : True
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeNetworkInterfaceAttribute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeNetworkInterfaces与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeNetworkInterfaces。

CLI

AWS CLI

描述您的网络接口

此示例描述了您的所有网络接口。

命令:

```
aws ec2 describe-network-interfaces
```

输出：

```
{
  "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"
        }
    ]
}
```

此示例描述了带有带有密钥Purpose和值的标签的网络接口Prod。

命令:

```
aws ec2 describe-network-interfaces --filters Name=tag:Purpose,Values=Prod
```

输出：

```
{
    "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"
        }
    ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeNetworkInterfaces](#)中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的网络接口。

```
Get-EC2NetworkInterface -NetworkInterfaceId eni-12345678
```

输出：

```
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
```

示例 2：此示例描述了您的所有网络接口。

```
Get-EC2NetworkInterface
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeNetworkInterfaces中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribePlacementGroups`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribePlacementGroups`。

CLI

AWS CLI

描述您的置放群组

此示例命令描述了您的所有置放群组。

命令:

```
aws ec2 describe-placement-groups
```

输出：

```
{
    "PlacementGroups": [
        {
            "GroupName": "my-cluster",
            "State": "available",
            "Strategy": "cluster"
        },
        ...
    ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribePlacementGroups中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的置放群组。

```
Get-EC2PlacementGroup -GroupName my-placement-group
```

输出：

```
GroupName             State        Strategy
---------             -----        --------
my-placement-group    available    cluster
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribePlacementGroups中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribePrefixLists`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribePrefixLists`。

CLI

AWS CLI

描述前缀列表

此示例列出了该区域的所有可用前缀列表。

命令:

```
aws ec2 describe-prefix-lists
```

输出：

```
{
  "PrefixLists": [
    {
      "PrefixListName": "com.amazonaws.us-east-1.s3",
      "Cidrs": [
        "54.231.0.0/17"
      ],
      "PrefixListId": "pl-63a5400a"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribePrefixLists中的。

PowerShell

用于 PowerShell

示例 1：此示例以前缀列表格式获取该区域的可用 AWS 服务 内容

```
Get-EC2PrefixList
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribePrefixLists中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeRegions与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeRegions。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::DescribeRegionsRequest request;
auto outcome = ec2Client.DescribeRegions(request);
bool result = true;
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;
        result = false;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考DescribeRegions中的。

CLI

AWS CLI

示例 1：描述所有已启用的区域

以下 describe-regions 示例描述了为您的账户启用的所有区域。

```
aws ec2 describe-regions
```

输出：

```
{
    "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"
        }
    ]
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的区域和区。

示例 2：描述具有端点的已启用区域，其名称包含特定字符串

以下 `describe-regions` 示例描述了在端点中具有字符串“us”的所有已启用区域。

```
aws ec2 describe-regions \
    --filters "Name=endpoint,Values=*us*"
```

输出：

```
{
    "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"
        }
    ]
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的区域和区。

示例 3：描述所有区域

以下 describe-regions 示例描述了所有可用区域，包括已禁用的区域。

```
aws ec2 describe-regions \
    --all-regions
```

输出：

```
{
    "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"
        }
    ]
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的区域和区。

示例 4：仅列出区域名称

以下 `describe-regions` 示例使用 `--query` 参数筛选输出，并仅以文本形式返回区域的名称。

```
aws ec2 describe-regions \
    --all-regions \
    --query "Regions[].{Name:RegionName}" \
    --output text
```

输出：

```
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
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的区域和区。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeRegions中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { DescribeRegionsCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

export const main = async () => {
  const command = new DescribeRegionsCommand({
    // By default this command will not show regions that require you to opt-in.
    // When AllRegions true even the regions that require opt-in will be
 returned.
    AllRegions: true,
    // You can omit the Filters property if you want to get all regions.
    Filters: [
      {
        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: ["ap-southeast-4"],
      },
    ],
  });

  try {
    const { Regions } = await client.send(command);
    const regionsList = Regions.map((reg) => ` • ${reg.RegionName}`);
    console.log("Found regions:");
    console.log(regionsList.join("\n"));
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DescribeRegions中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了可供您使用的区域。

```
Get-EC2Region
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DescribeRegions](#)中的。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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 "  "
    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__
```

- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考DescribeRegions中的。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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(())
}
```

- 有关 API 的详细信息，请参阅适用DescribeRegions于 Rust 的AWS SDK API 参考。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用DescribeRegions于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeRouteTables与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeRouteTables。

CLI

AWS CLI

描述您的路由表

以下describe-route-tables示例检索有关您的路由表的详细信息

```
aws ec2 describe-route-tables
```

输出：

```
{
    "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"
        }
    ]
}
```

有关更多信息，请参阅《AWS VPC 用户指南》中的使用路由表。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeRouteTables中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了您的所有路由表。

```
Get-EC2RouteTable
```

输出：

```
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  :
```

示例 2：此示例返回指定路由表的详细信息。

```
Get-EC2RouteTable -RouteTableId rtb-1a2b3c4d
```

示例 3：此示例描述了指定 VPC 的路由表。

```
Get-EC2RouteTable -Filter @{ Name="vpc-id"; Values="vpc-1a2b3c4d" }
```

输出：

```
Associations    : {rtbassoc-12345678}
```

```
PropagatingVgws : {}
Routes          : {, }
RouteTableId    : rtb-1a2b3c4d
Tags            : {}
VpcId           : vpc-1a2b3c4d
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeRouteTables中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeScheduledInstanceAvailability`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeScheduledInstanceAvailability`。

CLI

AWS CLI

描述可用的日程安排

此示例描述了从指定日期开始每周星期日发生的日程安排。

命令:

```
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
```

输出：

```
{
  "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"
    },
    ...
  ]
}
```

要缩小结果范围，您可以添加过滤器来指定操作系统、网络和实例类型。

命令:

--filters name=Platform，values=linux/UNIX Name=Network-Platform，values=ec2-vpc 名称=实例类型，values=c4.large

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeScheduledInstanceAvailability中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了从指定日期开始每周星期日发生的日程安排。

```
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
```

输出：

```
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

...
```

示例 2：要缩小结果范围，您可以为操作系统、网络和实例类型等条件添加筛选条件。

```
-Filter @{ Name="platform";Values="Linux/UNIX" },@{ Name="network-
platform";Values="EC2-VPC" },@{ Name="instance-type";Values="c4.large" }
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeScheduledInstanceAvailability中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeScheduledInstances与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeScheduledInstances。

CLI

AWS CLI

描述您的计划实例

此示例描述了指定的计划实例。

命令:

```
aws ec2 describe-scheduled-instances --scheduled-instance-ids
 sci-1234-1234-1234-1234-123456789012
```

输出：

```
{
  "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"
      }
  ]
}
```

此示例描述了您的所有计划实例。

命令:

```
aws ec2 describe-scheduled-instances
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeScheduledInstances中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的计划实例。

```
Get-EC2ScheduledInstance -ScheduledInstanceId
 sci-1234-1234-1234-1234-123456789012
```

输出：

```
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
```

示例 2：此示例描述了您的所有计划实例。

```
Get-EC2ScheduledInstance
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeScheduledInstances中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeSecurityGroups`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeSecurityGroups`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用实例

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
/// <summary>
/// Retrieve information for an Amazon EC2 security group.
/// </summary>
/// <param name="groupId">The Id of the Amazon EC2 security group.</param>
/// <returns>A list of security group information.</returns>
public async Task<List<SecurityGroup>> DescribeSecurityGroups(string groupId)
{
    var request = new DescribeSecurityGroupsRequest();
    var groupIds = new List<string> { groupId };
    request.GroupIds = groupIds;

    var response = await _amazonEC2.DescribeSecurityGroupsAsync(request);
    return response.SecurityGroups;
}

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

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考[DescribeSecurityGroups](#)中的。

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
##############################################################################
# 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 getopts 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
}
```

本示例中使用的实用程序函数。

```
###############################################################################
# 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
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSecurityGroups](#)中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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);
    }

    auto 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());
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考DescribeSecurityGroups中的。

CLI

AWS CLI

示例 1：描述安全组

以下 `describe-security-groups` 示例描述了指定的安全组。

```
aws ec2 describe-security-groups \
    --group-ids sg-903004f8
```

输出：

```
{
    "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",
        }
    ]
}
```

示例 2：描述具有特定规则的安全组

以下describe-security-groups示例使用过滤器将结果限定为具有允许 SSH 流量的规则（端口 22）和允许来自所有地址的流量的规则（0.0.0.0/0）的安全组。示例使用 --query

参数仅显示安全组的名称。安全组必须匹配要在结果中返回的所有筛选条件；但是，单个规则不必匹配所有筛选条件。例如，输出返回一个安全组，其中一个规则允许来自特定 IP 地址的 SSH 流量，另一个规则允许来自所有地址的 HTTP 流量。

```
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
```

输出：

```
default
my-security-group
web-servers
launch-wizard-1
```

示例 3：根据标签描述安全组

以下 `describe-security-groups` 示例使用筛选器，将结果范围限定为安全组名称中包含 `test` 且带有标签 `Test=To-delete` 的安全组。示例使用 `--query` 参数仅显示安全组的名称和 ID。

```
aws ec2 describe-security-groups \
    --filters Name=group-name,Values=*test* Name=tag:Test,Values=To-delete \
    --query "SecurityGroups[*].{Name:GroupName,ID:GroupId}"
```

输出：

```
[
    {
        "Name": "testfornewinstance",
        "ID": "sg-33bb22aa"
    },
    {
        "Name": "newgrouptest",
        "ID": "sg-1a2b3c4d"
    }
]
```

有关使用标签筛选器的更多示例，请参阅《Amazon EC2 用户指南》中的使用标签。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeSecurityGroups中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
   public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
       try {
           DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
               .groupIds(groupId)
               .build();

           // Use a paginator.
           DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
           listGroups.stream()
               .flatMap(r -> r.securityGroups().stream())
               .forEach(group -> System.out
                   .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));

       } catch (Ec2Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DescribeSecurityGroups中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

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

import { client } from "../libs/client.js";

// Log the details of a specific security group.
export const main = async () => {
  const command = new DescribeSecurityGroupsCommand({
    GroupIds: ["SECURITY_GROUP_ID"],
  });

  try {
    const { SecurityGroups } = await client.send(command);
    console.log(JSON.stringify(SecurityGroups, null, 2));
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DescribeSecurityGroups中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅适用DescribeSecurityGroups于 K otlin 的AWS SDK API 参考。

PowerShell

用于 PowerShell

示例 1：此示例描述了 VPC 的指定安全组。使用属于 VPC 的安全组时，必须使用安全组 ID（-GroupId 参数），而不是名称（-GroupName 参数）来引用该组。

```
Get-EC2SecurityGroup -GroupId sg-12345678
```

输出：

```
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
```

示例 2：此示例描述了 EC2-Classic 的指定安全组。使用 EC2-Classic 的安全组时，您可以使用组名称（-GroupName 参数）或组 ID（-GroupId 参数）来引用安全组。

```
Get-EC2SecurityGroup -GroupName my-security-group
```

输出：

```
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               :
```

示例 3：此示例检索 vpc-0fc1ff23456b789eb 的所有安全组

```
Get-EC2SecurityGroup -Filter @{Name="vpc-id";Values="vpc-0fc1ff23456b789eb"}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DescribeSecurityGroups](#)中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_resource, security_group=None):
        """
```

```
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param security_group: A Boto3 SecurityGroup object. This is a high-level
object
                               that wraps security group actions.
        """
        self.ec2_resource = ec2_resource
        self.security_group = security_group

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource)


    def describe(self):
        """
        Displays information about the security group.
        """
        if self.security_group is None:
            logger.info("No security group to describe.")
            return

        try:
            print(f"Security group: {self.security_group.group_name}")
            print(f"\tID: {self.security_group.id}")
            print(f"\tVPC: {self.security_group.vpc_id}")
            if self.security_group.ip_permissions:
                print(f"Inbound permissions:")
                pp(self.security_group.ip_permissions)
        except ClientError as err:
            logger.error(
                "Couldn't get data for security group %s. Here's why: %s: %s",
                self.security_group.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- 有关 API 的详细信息，请参阅适用DescribeSecurityGroups于 Python 的AWS SDK (Boto3) API 参考。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用DescribeSecurityGroups于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeSnapshotAttribute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeSnapshotAttribute`。

CLI

AWS CLI

描述快照的快照属性

以下describe-snapshot-attribute示例列出了与之共享快照的账户。

```
aws ec2 describe-snapshot-attribute \
    --snapshot-id snap-01234567890abcedf \
    --attribute createVolumePermission
```

输出：

```
{
    "SnapshotId": "snap-01234567890abcedf",
    "CreateVolumePermissions": [
        {
            "UserId": "123456789012"
        }
    ]
}
```

有关更多信息，请参阅亚马逊弹性计算云用户指南中的共享 Amazon EBS 快照。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeSnapshotAttribute中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定快照的指定属性。

```
Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute ProductCodes
```

输出：

```
CreateVolumePermissions    ProductCodes    SnapshotId
-----------------------    ------------    ----------
{}                         {}              snap-12345678
```

示例 2：此示例描述了指定快照的指定属性。

```
(Get-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission).CreateVolumePermissions
```

输出：

```
Group    UserId
-----    ------
all
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DescribeSnapshotAttribute](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeSnapshots`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeSnapshots`。

CLI

AWS CLI

示例 1：描述快照

以下 `describe-snapshots` 示例描述了指定的快照。

```
aws ec2 describe-snapshots \
    --snapshot-ids snap-1234567890abcdef0
```

输出：

```
{
    "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"
                }
            ]
        }
    ]
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的 Amazon EBS 快照。

示例 2：描述基于筛选器的快照

以下describe-snapshots示例使用筛选条件将结果范围限定为您的 AWS 账户拥有的处于该pending状态的快照。示例使用 --query 参数仅显示快照 ID 和快照的启动时间。

```
aws ec2 describe-snapshots \
    --owner-ids self \
    --filters Name=status,Values=pending \
    --query "Snapshots[*].{ID:SnapshotId,Time:StartTime}"
```

输出：

```
[
    {
        "ID": "snap-1234567890abcdef0",
        "Time": "2019-08-04T12:48:18.000Z"
    },
    {
        "ID": "snap-066877671789bd71b",
        "Time": "2019-08-04T02:45:16.000Z
    },
    ...
]
```

以下 describe-snapshots 示例使用筛选器，将结果范围限定为从指定卷创建的快照。示例使用 --query 参数仅显示快照 ID。

```
aws ec2 describe-snapshots \
    --filters Name=volume-id,Values=049df61146c4d7901 \
    --query "Snapshots[*].[SnapshotId]" \
    --output text
```

输出：

```
snap-1234567890abcdef0
snap-08637175a712c3fb9
...
```

有关使用筛选器的其他示例，请参阅《Amazon EC2 用户指南》中的列出和筛选资源。

示例 3：根据标签描述快照

以下 describe-snapshots 示例使用标签筛选器，将结果范围限定为带有标签 Stack=Prod 的快照。

```
aws ec2 describe-snapshots \
    --filters Name=tag:Stack,Values=prod
```

有关 describe-snapshots 的输出示例，请参阅示例 1。

有关使用标签筛选器的更多示例，请参阅《Amazon EC2 用户指南》中的使用标签。

示例 4：根据期限描述快照

以下describe-snapshots示例使用 JMESPath 表达式来描述您的 AWS 账户在指定日期之前创建的所有快照。仅显示快照 ID。

```
aws ec2 describe-snapshots \
    --owner-ids 012345678910 \
    --query "Snapshots[?(StartTime<='2020-03-31')].[SnapshotId]"
```

有关使用筛选器的其他示例，请参阅《Amazon EC2 用户指南》中的列出和筛选资源。

示例 5：仅查看存档的快照

以下 describe-snapshots 示例列出归档层中存储的快照。

```
aws ec2 describe-snapshots \
    --filters "Name=storage-tier,Values=archive"
```

输出：

```
{
    "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": []
        },
    ]
}
```

有关更多信息，请参阅《Amazon Elastic Compute Cloud 用户指南》中的查看已存档快照。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeSnapshots中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的快照。

```
Get-EC2Snapshot -SnapshotId snap-12345678
```

输出：

```
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
```

示例 2：此示例描述了带有 “名称” 标签的快照。

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" }
```

示例 3：此示例描述了带有 “名称” 标签且值为 “TestValue” 的快照。

```
Get-EC2Snapshot | ? { $_.Tags.Count -gt 0 -and $_.Tags.Key -eq "Name" -and
 $_.Tags.Value -eq "TestValue" }
```

示例 4：此示例描述了您的所有快照。

```
Get-EC2Snapshot -Owner self
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DescribeSnapshots](#)中的。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

显示快照的状态。

```
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(())
}
```

- 有关 API 的详细信息，请参阅适用DescribeSnapshots于 Rust 的AWS SDK API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeSpotDatafeedSubscription`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeSpotDatafeedSubscription`。

CLI

AWS CLI

描述账户的竞价型实例数据源订阅

此示例命令描述了账户的数据馈送。

命令:

```
aws ec2 describe-spot-datafeed-subscription
```

输出：

```
{
    "SpotDatafeedSubscription": {
        "OwnerId": "123456789012",
        "Prefix": "spotdata",
        "Bucket": "my-s3-bucket",
        "State": "Active"
    }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeSpotDatafeedSubscription中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了您的竞价型实例数据源。

```
Get-EC2SpotDatafeedSubscription
```

输出：

```
Bucket  : my-s3-bucket
Fault   :
OwnerId : 123456789012
Prefix  : spotdata
State   : Active
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeSpotDatafeedSubscription中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeSpotFleetInstances与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeSpotFleetInstances。

CLI

AWS CLI

描述与竞价型队列关联的竞价型实例

此示例命令列出了与指定竞价型队列关联的竞价型实例。

命令:

```
aws ec2 describe-spot-fleet-instances --spot-fleet-request-id sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

输出：

```
{
  "ActiveInstances": [
      {
          "InstanceId": "i-1234567890abcdef0",
          "InstanceType": "m3.medium",
          "SpotInstanceRequestId": "sir-08b93456"
```

```
        },
        ...
    ],
    "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeSpotFleetInstances中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了与指定竞价型队列请求关联的实例。

```
Get-EC2SpotFleetInstance -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

输出：

```
InstanceId    InstanceType    SpotInstanceRequestId
----------    ------------    ---------------------
i-f089262a    c3.large        sir-12345678
i-7e8b24a4    c3.large        sir-87654321
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeSpotFleetInstances中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeSpotFleetRequestHistory与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeSpotFleetRequestHistory。

CLI

AWS CLI

描述 Spot 舰队的历史

此示例命令返回从指定时间开始的指定 Spot 队列的历史记录。

命令:

```
aws ec2 describe-spot-fleet-request-history --spot-fleet-request-id sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE --start-time 2015-05-26T00:00:00Z
```

以下示例输出显示了竞价型队列的两个竞价型实例的成功启动。

输出：

```
{
  "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+TfGlWxWWB/Vr5dk5d4LfdgA/DRAHUrYgxzrEXAMPLE=",
  "StartTime": "2015-05-26T00:00:00Z"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeSpotFleetRequestHistory中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定 Spot 队列请求的历史记录。

```
Get-EC2SpotFleetRequestHistory -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -StartTime 2015-12-26T00:00:00Z
```

输出：

```
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
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeSpotFleetRequestHistory中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeSpotFleetRequests`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeSpotFleetRequests`。

CLI

AWS CLI

描述您的 Spot 队列请求

此示例描述了您的所有 Spot 队列请求。

命令:

```
aws ec2 describe-spot-fleet-requests
```

输出：

```
{
  "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"
    }
```

```
  ]
}
```

描述 Spot 队列请求

此示例描述了指定的 Spot 队列请求。

命令:

```
aws ec2 describe-spot-fleet-requests --spot-fleet-request-ids sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

输出：

```
{
  "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"
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSpotFleetRequests](#)中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的 Spot 队列请求。

```
Get-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE | format-list
```

输出：

```
ConfigData            : Amazon.EC2.Model.SpotFleetRequestConfigData
CreateTime            : 12/26/2015 8:23:33 AM
SpotFleetRequestId    : sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
SpotFleetRequestState : active
```

示例 2：此示例描述了您的所有竞价型队列请求。

```
Get-EC2SpotFleetRequest
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DescribeSpotFleetRequests](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeSpotInstanceRequests`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeSpotInstanceRequests`。

CLI

AWS CLI

示例 1：描述竞价型实例请求

以下`describe-spot-instance-requests`示例描述了指定的竞价型实例请求。

```
aws ec2 describe-spot-instance-requests \
    --spot-instance-request-ids sir-08b93456
```

输出：

```
{
    "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"
        }
    ]
}
```

示例 2：描述基于筛选条件的竞价型实例请求

以下describe-spot-instance-requests示例使用筛选条件将结果限定为指定可用区内具有指定实例类型的竞价型实例请求。该示例使用--query参数仅显示实例 ID。

```
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
```

输出：

```
i-057750d42936e468a
i-001efd250faaa6ffa
i-027552a73f021f3bd
...
```

有关使用筛选条件的其他示例，请参阅亚马逊弹性计算云用户指南中的列出和筛选您的资源。

示例 3：描述基于标签的竞价型实例请求

以下describe-spot-instance-requests示例使用标签筛选器将结果范围限定为带有该标签的竞价型实例请求cost-center=cc123。

```
aws ec2 describe-spot-instance-requests \
    --filters Name=tag:cost-center,Values=cc123
```

有关 describe-spot-instance-requests 的输出示例，请参阅示例 1。

有关使用标签筛选器的更多示例，请参阅《Amazon EC2 用户指南》中的使用标签。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeSpotInstanceRequests中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的竞价型实例请求。

```
Get-EC2SpotInstanceRequest -SpotInstanceRequestId sir-12345678
```

输出：

```
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
```

示例 2：此示例描述了您的所有竞价型实例请求。

```
Get-EC2SpotInstanceRequest
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeSpotInstanceRequests中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeSpotPriceHistory`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeSpotPriceHistory`。

CLI

AWS CLI

描述现货价格历史记录

此示例命令返回 m1.xlarge 实例在一月份特定日期的竞价价格历史记录。

命令:

```
aws ec2 describe-spot-price-history --instance-types m1.xlarge --start-time
 2014-01-06T07:08:09 --end-time 2014-01-06T08:09:10
```

输出：

```
{
  "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"
      },
      ...
}
```

描述 Linux/UNIX 亚马逊 VPC 的现货价格历史记录

此示例命令返回 m1.xlarge、Linux/UNIX Amazon VPC 实例在一月份特定日期的现货价格历史记录。

命令:

```
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
```

输出：

```
{
  "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"
        }
    ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[DescribeSpotPriceHistory](#)中的。

PowerShell

用于 PowerShell

示例 1：此示例获取指定实例类型和可用区域的竞价价格历史记录中的最后 10 个条目。请注意，为-AvailabilityZone 参数指定的值必须对提供给 cmdlet 的-Region 参数（示例中未显示）的区域值有效，或者在 shell 中设置为默认值。此示例命令假设环境中已设置默认区域 “us-west-2”。

```
Get-EC2SpotPriceHistory -InstanceType c3.large -AvailabilityZone us-west-2a -
MaxResult 10
```

输出：

```
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
...
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DescribeSpotPriceHistory](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeSubnets`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeSubnets`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- [构建和管理弹性服务](#)

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#)中查找完整实例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考DescribeSubnets中的。

CLI

AWS CLI

示例 1：描述所有子网

以下 `describe-subnets` 示例显示子网的详细信息。

```
aws ec2 describe-subnets
```

输出：

```
{
    "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
            }
        }
    ]
}
```

有关更多信息，请参阅《AWS VPC 用户指南》中的使用 VPC 和子网。

示例 2：描述特定 VPC 的子网

以下 describe-subnets 示例使用筛选条件检索指定 VPC 的子网的详细信息。

```
aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=vpc-3EXAMPLE"
```

输出：

```
{
    "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
            }
        }
    ]
}
```

有关更多信息，请参阅《AWS VPC 用户指南》中的使用 VPC 和子网。

示例 3：描述带有特定标签的子网

以下 `describe-subnets` 示例使用筛选器检索带有标签 `CostCenter=123` 的子网的详细信息，并使用 `--query` 参数显示带有此标签子网的子网 ID。

```
aws ec2 describe-subnets \
    --filters "Name=tag:CostCenter,Values=123" \
    --query "Subnets[*].SubnetId" \
    --output text
```

输出：

```
subnet-0987a87c8b37348ef
subnet-02a95061c45f372ee
subnet-03f720e7de2788d73
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的使用 VPC 和子网。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeSubnets中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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"] },
    ],
  }),
);
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DescribeSubnets中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的子网。

```
Get-EC2Subnet -SubnetId subnet-1a2b3c4d
```

输出：

```
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
```

示例 2：此示例描述了您的所有子网。

```
Get-EC2Subnet
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DescribeSubnets](#)中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class AutoScaler:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :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
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"
        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"
        self.key_pair_name = f"{resource_prefix}-key-pair"


    def get_subnets(self, vpc_id, zones):
        """
        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.
        """
        try:
            response = self.ec2_client.describe_subnets(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )
            subnets = response["Subnets"]
            log.info("Found %s subnets for the specified zones.", len(subnets))
        except ClientError as err:
            raise AutoScalerError(f"Couldn't get subnets: {err}")
```

```
        else:
            return subnets
```

- 有关 API 的详细信息，请参阅适用DescribeSubnets于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeTags`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeTags`。

CLI

AWS CLI

示例 1：描述单个资源的所有标签

以下`describe-tags`示例描述了指定实例的标签。

```
aws ec2 describe-tags \
    --filters "Name=resource-id,Values=i-1234567890abcdef8"
```

输出：

```
{
    "Tags": [
        {
            "ResourceType": "instance",
            "ResourceId": "i-1234567890abcdef8",
            "Value": "Test",
            "Key": "Stack"
        },
        {
            "ResourceType": "instance",
            "ResourceId": "i-1234567890abcdef8",
            "Value": "Beta Server",
            "Key": "Name"
        }
```

```
    ]
}
```

示例 2：描述资源类型的所有标签

以下describe-tags示例描述了您的卷的标签。

```
aws ec2 describe-tags \
    --filters "Name=resource-type,Values=volume"
```

输出：

```
{
    "Tags": [
        {
            "ResourceType": "volume",
            "ResourceId": "vol-1234567890abcdef0",
            "Value": "Project1",
            "Key": "Purpose"
        },
        {
            "ResourceType": "volume",
            "ResourceId": "vol-049df61146c4d7901",
            "Value": "Logs",
            "Key": "Purpose"
        }
    ]
}
```

示例 3：描述您的所有标签

以下describe-tags示例描述了您的所有资源的标签。

```
aws ec2 describe-tags
```

示例 4：根据标签密钥描述资源的标签

以下describe-tags示例描述了带有密钥标签的资源的标签Stack。

```
aws ec2 describe-tags \
    --filters Name=key,Values=Stack
```

输出：

```
{
    "Tags": [
        {
            "ResourceType": "volume",
            "ResourceId": "vol-027552a73f021f3b",
            "Value": "Production",
            "Key": "Stack"
        },
        {
            "ResourceType": "instance",
            "ResourceId": "i-1234567890abcdef8",
            "Value": "Test",
            "Key": "Stack"
        }
    ]
}
```

示例 5：根据标签键和标签值描述资源的标签

以下`describe-tags`示例描述了带有该标签的资源的标签`Stack=Test`。

```
aws ec2 describe-tags \
    --filters Name=key,Values=Stack Name=value,Values=Test
```

输出：

```
{
    "Tags": [
        {
            "ResourceType": "image",
            "ResourceId": "ami-3ac336533f021f3bd",
            "Value": "Test",
            "Key": "Stack"
        },
        {
            "ResourceType": "instance",
            "ResourceId": "i-1234567890abcdef8",
            "Value": "Test",
            "Key": "Stack"
        }
```

```
    ]
}
```

以下describe-tags示例使用替代语法来描述带有标签的资源Stack=Test。

```
aws ec2 describe-tags \
    --filters "Name=tag:Stack,Values=Test"
```

以下describe-tags示例描述了所有带有密钥但没有值的标签的实例Purpose的标签。

```
aws ec2 describe-tags \
    --filters "Name=resource-type,Values=instance" "Name=key,Values=Purpose"
 "Name=value,Values="
```

输出：

```
{
    "Tags": [
        {
            "ResourceType": "instance",
            "ResourceId": "i-1234567890abcdef5",
            "Value": null,
            "Key": "Purpose"
        }
    ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeTags中的。

PowerShell

用于 PowerShell

示例 1：此示例获取资源类型 “image” 的标签

```
Get-EC2Tag -Filter @{Name="resource-type";Values="image"}
```

输出：

```
Key         ResourceId            ResourceType Value
```

```
---          ----------            ------------ -----
Name         ami-0a123b4ccb567a8ea image        Win7-Imported
auto-delete ami-0a123b4ccb567a8ea image        never
```

示例 2：此示例提取所有资源的所有标签并按资源类型对它们进行分组

```
Get-EC2Tag | Group-Object resourcetype
```

输出：

```
Count Name                      Group
----- ----                      -----
    9 subnet                    {Amazon.EC2.Model.TagDescription,
 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,
 Amazon.EC2.Model.TagDescription...}
    3 route-table               {Amazon.EC2.Model.TagDescription,
 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}
```

示例 3：此示例显示了给定区域中所有带有 “自动删除” 标签且值为 “否” 的资源

```
Get-EC2Tag -Region eu-west-1 -Filter @{Name="tag:auto-delete";Values="no"}
```

输出：

```
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
```

示例 4：此示例获取所有带有 “auto-delete” 标签且值为 “无” 的资源，并在下一个管道中进行进一步筛选以仅解析 “实例” 资源类型，最终为每个实例资源创建 ThisInstance “” 标签，其值为实例 ID 本身

```
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}}
```

示例 5：此示例获取所有实例资源的标签以及 “名称” 密钥并以表格格式显示它们

```
Get-EC2Tag -Filter @{Name="resource-
type";Values="instance"},@{Name="key";Values="Name"} | Select-Object ResourceId,
 @{Name="Name-Tag";Expression={$PSItem.Value}} | Format-Table -AutoSize
```

输出：

```
ResourceId          Name-Tag
----------          --------
i-012e3cb4df567e1aa jump1
i-01c23a45d6fc7a89f repro-3
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DescribeTags](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeVolumeAttribute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeVolumeAttribute`。

CLI

AWS CLI

描述音量属性

此示例命令描述带有 ID 的卷的autoEnableIo属性vol-049df61146c4d7901。

命令:

```
aws ec2 describe-volume-attribute --volume-id vol-049df61146c4d7901 --attribute
 autoEnableIO
```

输出：

```
{
    "AutoEnableIO": {
        "Value": false
    },
    "VolumeId": "vol-049df61146c4d7901"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeVolumeAttribute中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定卷的指定属性。

```
Get-EC2VolumeAttribute -VolumeId vol-12345678 -Attribute AutoEnableIO
```

输出：

```
AutoEnableIO    ProductCodes    VolumeId
------------    ------------    --------
False           {}              vol-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeVolumeAttribute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeVolumeStatus`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeVolumeStatus`。

CLI

AWS CLI

描述单个卷的状态

此示例命令描述了卷的状态`vol-1234567890abcdef0`。

命令:

```
aws ec2 describe-volume-status --volume-ids vol-1234567890abcdef0
```

输出：

```
{
    "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": []
        }
```

```
    ]
}
```

描述受损卷的状态

此示例命令描述了所有受损卷的状态。在此示例输出中，没有受损的音量。

命令:

```
aws ec2 describe-volume-status --filters Name=volume-
status.status,Values=impaired
```

输出：

```
{
    "VolumeStatuses": []
}
```

如果您的卷状态检查失败（状态受损），请参阅 Amazon EC2 用户指南中的处理受损卷。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeVolumeStatus中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定卷的状态。

```
Get-EC2VolumeStatus -VolumeId vol-12345678
```

输出：

```
Actions          : {}
AvailabilityZone : us-west-2a
Events           : {}
VolumeId         : vol-12345678
VolumeStatus     : Amazon.EC2.Model.VolumeStatusInfo
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus
```

输出：

```
Details                         Status
-------                         ------
{io-enabled, io-performance}    ok
```

```
(Get-EC2VolumeStatus -VolumeId vol-12345678).VolumeStatus.Details
```

输出：

```
Name                            Status
----                            ------
io-enabled                      passed
io-performance                  not-applicable
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DescribeVolumeStatus](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeVolumes`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeVolumes`。

CLI

AWS CLI

示例 1：描述卷

以下`describe-volumes`示例描述了当前区域中的指定卷。

```
aws ec2 describe-volumes \
    --volume-ids vol-049df61146c4d7901 vol-1234567890abcdef0
```

输出：

```
{
    "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
        }
    ]
}
```

示例 2：描述连接到特定实例的卷

以下describe-volumes示例描述了所有既连接到指定实例又设置为在实例终止时删除的卷。

```
aws ec2 describe-volumes \
    --region us-east-1 \
```

```
    --filters Name=attachment.instance-id,Values=i-1234567890abcdef0
 Name=attachment.delete-on-termination,Values=true
```

有关 describe-volumes 的输出示例，请参阅示例 1。

示例 3：描述特定可用区中的可用卷

以下describe-volumes示例描述了状态为available且位于指定可用区的所有卷。

```
aws ec2 describe-volumes \
    --filters Name=status,Values=available Name=availability-zone,Values=us-
east-1a
```

有关 describe-volumes 的输出示例，请参阅示例 1。

示例 4：根据标签描述卷

以下describe-volumes示例描述了所有具有标签键Name和以开头的值的卷Test。然后，使用仅显示卷标签和 ID 的查询筛选输出。

```
aws ec2 describe-volumes \
    --filters Name=tag:Name,Values=Test* \
    --query "Volumes[*].{ID:VolumeId,Tag:Tags}"
```

输出：

```
[
    {
       "Tag": [
           {
               "Value": "Test2",
               "Key": "Name"
           }
       ],
       "ID": "vol-1234567890abcdef0"
   },
   {
       "Tag": [
           {
               "Value": "Test1",
               "Key": "Name"
           }
       ],
```

```
        "ID": "vol-049df61146c4d7901"
    }
]
```

有关使用标签筛选器的更多示例，请参阅《Amazon EC2 用户指南》中的使用标签。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeVolumes中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的 EBS 卷。

```
Get-EC2Volume -VolumeId vol-12345678
```

输出：

```
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
```

示例 2：此示例描述了状态为 “可用” 的 EBS 卷。

```
Get-EC2Volume -Filter @{ Name="status"; Values="available" }
```

输出：

```
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
...
```

示例 3：此示例描述了您的所有 EBS 卷。

```
Get-EC2Volume
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DescribeVolumes](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeVpcAttribute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeVpcAttribute`。

CLI

AWS CLI

描述该 enableDnsSupport 属性

此示例描述了该`enableDnsSupport`属性。此属性指示 VPC 是否已启用 DNS 解析。如果该属性为 `true`，则 Amazon DNS 服务器会将您实例的 DNS 主机名称解析为相应的 IP 地址，否则不会解析。

命令:

```
aws ec2 describe-vpc-attribute --vpc-id vpc-a01106c2 --attribute enableDnsSupport
```

输出：

```
{
    "VpcId": "vpc-a01106c2",
```

```
    "EnableDnsSupport": {
        "Value": true
    }
}
```

描述该 enableDnsHostnames 属性

此示例描述了该enableDnsHostnames属性。此属性表示在 VPC 中启动的实例是否获得 DNS 主机名。如果该属性为 true，则 VPC 内的实例可获得 DNS 主机名称，否则将无法获得。

命令:

```
aws ec2 describe-vpc-attribute --vpc-id vpc-a01106c2 --attribute
 enableDnsHostnames
```

输出：

```
{
    "VpcId": "vpc-a01106c2",
    "EnableDnsHostnames": {
        "Value": true
    }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeVpcAttribute中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了 “enableDnsSupport” 属性。

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsSupport
```

输出：

```
EnableDnsSupport
----------------
True
```

示例 2：此示例描述了 “enableDnsHostnames” 属性。

```
Get-EC2VpcAttribute -VpcId vpc-12345678 -Attribute enableDnsHostnames
```

输出：

```
EnableDnsHostnames
------------------
True
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeVpcAttribute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeVpcClassicLink与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeVpcClassicLink。

CLI

AWS CLI

描述您的 VPC 的 ClassicLink 状态

此示例列出了 vpc- ClassicLink 88888888 的状态。

命令:

```
aws ec2 describe-vpc-classic-link --vpc-id vpc-88888888
```

输出：

```
{
  "Vpcs": [
    {
      "ClassicLinkEnabled": true,
      "VpcId": "vpc-88888888",
      "Tags": [
        {
          "Value": "classiclinkvpc",
```

```
          "Key": "Name"
        }
      ]
    }
  ]
}
```

此示例仅列出已启用 Classiclink 的 VPC（的筛选值设置`is-classic-link-enabled`为）。`true`

命令:

```
aws ec2 describe-vpc-classic-link --filter "Name=is-classic-link-
enabled,Values=true"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeVpcClassicLink中的。

PowerShell

用于 PowerShell

示例 1：上面的示例返回了所有 VPC 及其在该 ClassicLinkEnabled 区域的状态

```
Get-EC2VpcClassicLink -Region eu-west-1
```

输出：

```
ClassicLinkEnabled Tags   VpcId
------------------ ----   -----
False              {Name} vpc-0fc1ff23f45b678eb
False              {}     vpc-01e23c4a5d6db78e9
False              {Name} vpc-0123456b078b9d01f
False              {}     vpc-12cf3b4f
False              {Name} vpc-0b12d3456a7e8901d
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeVpcClassicLink中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeVpcClassicLinkDnsSupport与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeVpcClassicLinkDnsSupport。

CLI

AWS CLI

描述您的 VPC 的 ClassicLink DNS 支持

此示例描述了您的所有 VPC 的 ClassicLink DNS 支持状态。

命令:

```
aws ec2 describe-vpc-classic-link-dns-support
```

输出：

```
{
  "Vpcs": [
    {
      "VpcId": "vpc-88888888",
      "ClassicLinkDnsSupported": true
    },
    {
      "VpcId": "vpc-1a2b3c4d",
      "ClassicLinkDnsSupported": false
    }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeVpcClassicLinkDnsSupport中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了 eu-west-1 区域 VPC 的 ClassicLink DNS 支持状态

```
Get-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

输出：

```
ClassicLinkDnsSupported VpcId
----------------------- -----
False                   vpc-0b12d3456a7e8910d
False                   vpc-12cf3b4f
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DescribeVpcClassicLinkDnsSupport](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeVpcEndpointServices`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeVpcEndpointServices`。

CLI

AWS CLI

示例 1：描述所有 VPC 终端节点服务

以下 “describe-vpc-endpoint-services” 示例列出了某个 AWS 区域的所有 VPC 终端节点服务。

```
aws ec2 describe-vpc-endpoint-services
```

输出：

```
{
    "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"
    ]
}
```

有关更多信息，请参阅《用户指南》中的查看可用 AWS 服务名称 AWS PrivateLink。

示例 2：描述终端节点服务的详细信息

以下 “describe-vpc-endpoint-services” 示例列出了 Amazon S3 接口终端节点服务的详细信息

```
aws ec2 describe-vpc-endpoint-services \
    --filter "Name=service-type,Values=Interface" Name=service-
name,Values=com.amazonaws.us-east-1.s3
```

输出：

```
{
    "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"
    ]
}
```

有关更多信息，请参阅《用户指南》中的查看可用 AWS 服务名称 AWS PrivateLink。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeVpcEndpointServices中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了使用给定筛选器的 EC2 VPC 终端节点服务，在本例中为 com.amazonaws.eu-west-1.ecs。此外，它还会扩展 ServiceDetails 属性并显示细节

```
Get-EC2VpcEndpointService -Region eu-west-1 -MaxResult 5 -Filter @{Name="service-
name";Values="com.amazonaws.eu-west-1.ecs"} | Select-Object -ExpandProperty
 ServiceDetails
```

输出：

```
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
```

示例 2：此示例检索所有 EC2 VPC 终端节点服务并返回 ServiceNames 匹配的 “ssm”

```
Get-EC2VpcEndpointService -Region eu-west-1 | Select-Object -ExpandProperty
 Servicenames | Where-Object { -match "ssm"}
```

输出：

```
com.amazonaws.eu-west-1.ssm
com.amazonaws.eu-west-1.ssmmessages
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeVpcEndpointServices中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeVpcEndpoints`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeVpcEndpoints`。

CLI

AWS CLI

描述您的 VPC 终端节点

以下`describe-vpc-endpoints`示例显示了您的所有 VPC 终端节点的详细信息。

```
aws ec2 describe-vpc-endpoints
```

输出：


```
{
    "VpcEndpoints": [
        {
            "PolicyDocument": "{\"Version\":\"2008-10-17\",\"Statement\":
[{\"Effect\":\"Allow\",\"Principal\":\"*\",\"Action\":\"*\",\"Resource\":\"*
\"}]}",
            "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      \"Effect\": \"Allow\", \n      \"Principal\": \"*\", \n
 \"Resource\": \"*\"\n    }\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-111122223333aabbc",
            "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"
        }
    ]
```

```
}
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的 VPC 端点。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeVpcEndpoints中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了您的 eu-west-1 区域的一个或多个 VPC 终端节点。然后，它将输出通过管道传递给下一个命令，该命令选择 VpcEndpointId 属性并以字符串数组的形式返回数组 VPC ID

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object -ExpandProperty
 VpcEndpointId
```

输出：

```
vpce-01a2ab3f4f5cc6f7d
vpce-01d2b345a6787890b
vpce-0012e34d567890e12
vpce-0c123db4567890123
```

示例 2：此示例描述了 eu-west-1 区域的所有 vpc 终端节点，并 VpcEndpointId选择 VpcId ServiceName 、 PrivateDnsEnabled 和属性以表格格式呈现

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
 ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

输出：

```
VpcEndpointId          VpcId                ServiceName
 PrivateDnsEnabled
-------------          -----                -----------
 -----------------
vpce-02a2ab2f2f2cc2f2d vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssm
             True
vpce-01d1b111a1114561b vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2
             True
```

```
vpce-0011e23d45167e838 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ec2messages
             True
vpce-0c123db4567890123 vpc-0fc6ff46f65b039eb com.amazonaws.eu-west-1.ssmmessages
             True
```

示例 3：此示例将 VPC 终端节点 vpce-01a2ab3f4f5cc6f7d 的策略文档导出到 json 文件中

```
Get-EC2VpcEndpoint -Region eu-west-1 -VpcEndpointId vpce-01a2ab3f4f5cc6f7d |
 Select-Object -expand PolicyDocument | Out-File vpce_policyDocument.json
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeVpcEndpoints中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DescribeVpcs`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DescribeVpcs`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 构建和管理弹性服务

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
```

```
    {
        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];
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考DescribeVpcs中的。

CLI

AWS CLI

示例 1：描述所有 VPC

以下 `describe-vpcs` 示例检索 VPC 的详细信息。

```
aws ec2 describe-vpcs
```

输出：

```
{
    "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"
                }
            ]
        }
    ]
}
```

示例 2：描述指定的 VPC

以下 `describe-vpcs` 示例检索指定 VPC 的详细信息。

```
aws ec2 describe-vpcs \
    --vpc-ids vpc-06e4ab6c6cEXAMPLE
```

输出：

```
{
    "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"
                }
            ]
        }
    ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeVpcs中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DescribeVpcs中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的 VPC。

```
Get-EC2Vpc -VpcId vpc-12345678
```

输出：

```
CidrBlock       : 10.0.0.0/16
DhcpOptionsId   : dopt-1a2b3c4d
InstanceTenancy : default
IsDefault       : False
State           : available
Tags            : {Name}
VpcId           : vpc-12345678
```

示例 2：此示例描述了默认 VPC（每个区域只能有一个）。如果您的账户在此区域支持 EC2-Classic，则没有默认 VPC。

```
Get-EC2Vpc -Filter @{Name="isDefault"; Values="true"}
```

输出：

```
CidrBlock       : 172.31.0.0/16
DhcpOptionsId   : dopt-12345678
InstanceTenancy : default
IsDefault       : True
State           : available
Tags            : {}
```

```
VpcId           : vpc-45678901
```

示例 3：此示例描述了与指定筛选条件匹配的 VPC（即，其 CIDR 与值 “10.0.0.0/16” 匹配且状态为 “可用”）。

```
Get-EC2Vpc -Filter @{Name="cidr";
 Values="10.0.0.0/16"},@{Name="state";Values="available"}
```

示例 4：此示例描述了您的所有 VPC。

```
Get-EC2Vpc
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeVpcs中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class AutoScaler:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
```

```
        :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
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"
        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"
        self.key_pair_name = f"{resource_prefix}-key-pair"


    def get_default_vpc(self):
        """
        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:
            raise AutoScalerError(f"Couldn't get default VPC: {err}")
        else:
            return response["Vpcs"][0]
```

- 有关 API 的详细信息，请参阅适用DescribeVpcs于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeVpnConnections与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeVpnConnections。

CLI

AWS CLI

示例 1：描述您的 VPN 连接

以下describe-vpn-connections示例描述了您的所有站点到站点 VPN 连接。

```
aws ec2 describe-vpn-connections
```

输出：

```
{
    "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": ""
                }
            ]
        }
    ]
}
```

有关更多信息，请参阅站点到站点 VPN 用户指南中的 AWS 站点到站点 VPN 的工作原理。AWS

示例 2：描述您的可用的 VPN 连接

以下`describe-vpn-connections`示例描述了您的站点到站点 VPN 连接，其状态为。`available`

```
aws ec2 describe-vpn-connections \
    --filters "Name=state,Values=available"
```

有关更多信息，请参阅站点到站点 VPN 用户指南中的 AWS 站点到站点 VPN 的工作原理。AWS

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeVpnConnections中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的 VPN 连接。

```
Get-EC2VpnConnection -VpnConnectionId vpn-12345678
```

输出：

```
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
```

示例 2：此示例描述了状态为待处理或可用状态的任何 VPN 连接。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnConnection -Filter $filter
```

示例 3：此示例描述了您的所有 VPN 连接。

```
Get-EC2VpnConnection
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeVpnConnections中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DescribeVpnGateways与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DescribeVpnGateways。

CLI

AWS CLI

描述您的虚拟专用网关

此示例描述您的虚拟专用网关。

命令:

```
aws ec2 describe-vpn-gateways
```

输出：

```
{
    "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"
                }
            ]
        }
    ]
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DescribeVpnGateways中的。

PowerShell

用于 PowerShell

示例 1：此示例描述了指定的虚拟专用网关。

```
Get-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d
```

输出：

```
AvailabilityZone :
State            : available
Tags             : {}
Type             : ipsec.1
VpcAttachments   : {vpc-12345678}
VpnGatewayId     : vgw-1a2b3c4d
```

示例 2：此示例描述了状态为 “待定” 或 “可用” 的所有虚拟专用网关。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "state"
$filter.Values = @( "pending", "available" )

Get-EC2VpnGateway -Filter $filter
```

示例 3：此示例描述了您的所有虚拟专用网关。

```
Get-EC2VpnGateway
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DescribeVpnGateways中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DetachInternetGateway与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DetachInternetGateway。

CLI

AWS CLI

将互联网网关与您的 VPC 分离

以下detach-internet-gateway示例将指定的互联网网关与特定 VPC 分离。

```
aws ec2 detach-internet-gateway \
    --internet-gateway-id igw-0d0fb496b3EXAMPLE \
    --vpc-id vpc-0a60eb65b4EXAMPLE
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon VPC 用户指南》中的互联网网关。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DetachInternetGateway中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的互联网网关与指定 VPC 分离。

```
Dismount-EC2InternetGateway -InternetGatewayId igw-1a2b3c4d -VpcId vpc-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DetachInternetGateway中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DetachNetworkInterface与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DetachNetworkInterface。

CLI

AWS CLI

将网络接口与您的实例分离

此示例将指定的网络接口与指定实例分离。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 detach-network-interface --attachment-id eni-attach-66c4350a
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DetachNetworkInterface中的。

PowerShell

用于 PowerShell

示例 1：此示例删除网络接口和实例之间的指定连接。

```
Dismount-EC2NetworkInterface -AttachmentId eni-attach-1a2b3c4d -Force
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DetachNetworkInterface中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DetachVolume`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DetachVolume`。

CLI

AWS CLI

将卷与实例分离

此示例命令将卷 (`vol-049df61146c4d7901`) 与其连接的实例分离。

命令:

```
aws ec2 detach-volume --volume-id vol-1234567890abcdef0
```

输出：

```
{
    "AttachTime": "2014-02-27T19:23:06.000Z",
    "InstanceId": "i-1234567890abcdef0",
    "VolumeId": "vol-049df61146c4d7901",
    "State": "detaching",
    "Device": "/dev/sdb"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DetachVolume中的。

PowerShell

用于 PowerShell

示例 1：此示例分离指定的卷。

```
Dismount-EC2Volume -VolumeId vol-12345678
```

输出：

```
AttachTime          : 12/22/2015 1:53:58 AM
DeleteOnTermination : False
Device              : /dev/sdh
InstanceId          : i-1a2b3c4d
State               : detaching
VolumeId            : vol-12345678
```

示例 2：您还可以指定实例 ID 和设备名称，以确保分离的卷正确无误。

```
Dismount-EC2Volume -VolumeId vol-12345678 -InstanceId i-1a2b3c4d -Device /dev/sdh
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DetachVolume中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DetachVpnGateway与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DetachVpnGateway。

CLI

AWS CLI

将虚拟私有网关与您的 VPC 分离

此示例将指定的虚拟私有网关与指定 VPC 分离。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 detach-vpn-gateway --vpn-gateway-id vgw-9a4cacf3 --vpc-id vpc-a01106c2
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DetachVpnGateway中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的虚拟私有网关与指定 VPC 分离。

```
Dismount-EC2VpnGateway -VpnGatewayId vgw-1a2b3c4d -VpcId vpc-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DetachVpnGateway中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DisableVgwRoutePropagation与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DisableVgwRoutePropagation。

CLI

AWS CLI

禁用路由传播

此示例禁止指定的虚拟专用网关向指定路由表传播静态路由。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 disable-vgw-route-propagation --route-table-id rtb-22574640 --gateway-id
 vgw-9a4cacf3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DisableVgwRoutePropagation中的。

PowerShell

用于 PowerShell

示例 1：此示例禁止 VGW 自动将路由传播到指定的路由表。

```
Disable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DisableVgwRoutePropagation中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DisableVpcClassicLink`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DisableVpcClassicLink`。

CLI

AWS CLI

为 VPC 禁用 ClassicLink

此示例 ClassicLink 对于 vpc-8888888 禁用。

命令:

```
aws ec2 disable-vpc-classic-link --vpc-id vpc-88888888
```

输出：

```
{
  "Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DisableVpcClassicLink中的。

PowerShell

用于 PowerShell

示例 1：此示例禁用 vpc-01e23c4a5d6db VpcClassicLink 78e9 的 EC2。它返回 True 或 False

```
Disable-EC2VpcClassicLink -VpcId vpc-01e23c4a5d6db78e9
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DisableVpcClassicLink中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DisableVpcClassicLinkDnsSupport`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DisableVpcClassicLinkDnsSupport`。

CLI

AWS CLI

禁用 VPC 的 ClassicLink DNS 支持

此示例禁用对 ClassicLink 的 DNS 支持。`vpc-88888888`

命令:

```
aws ec2 disable-vpc-classic-link-dns-support --vpc-id vpc-88888888
```

输出：

```
{
  "Return": true
```

```
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DisableVpcClassicLinkDnsSupport中的。

PowerShell

用于 PowerShell

示例 1：此示例禁用了对 vpc-0b12d3456a7e ClassicLink 8910d 的 DNS 支持

```
Disable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DisableVpcClassicLinkDnsSupport中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DisassociateAddress`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DisassociateAddress`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用实例

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    /// <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)
    {
        var response = await _amazonEC2.DisassociateAddressAsync(
            new DisassociateAddressRequest { AssociationId = associationId });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考[DisassociateAddress](#)中的。

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
###############################################################################
# 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
}
```

本示例中使用的实用程序函数。

```
##############################################################################
# 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
}
```

```
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DisassociateAddress中的。

CLI

AWS CLI

解除关联 EC2-Classic 中的弹性 IP 地址

以下示例将弹性 IP 地址与 EC2 Classic 中的实例解除关联。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 disassociate-address --public-ip 198.51.100.0
```

解除关联 EC2-VPC 中的弹性 IP 地址

以下示例将弹性 IP 地址与 VPC 中的实例解除关联。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 disassociate-address --association-id eipassoc-2bebb745
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DisassociateAddress中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    public static void disassociateAddress(Ec2Client ec2, String associationId) {
```

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

            ec2.disassociateAddress(addressRequest);
            System.out.println("You successfully disassociated the address!");

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考DisassociateAddress中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { DisassociateAddressCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

// Disassociate an Elastic IP address from an instance.
export const main = async () => {
  const command = new DisassociateAddressCommand({
    // You can also use PublicIp, but that is for EC2 classic which is being
 retired.
    AssociationId: "ASSOCIATION_ID",
  });

  try {
    await client.send(command);
```

```
    console.log("Successfully disassociated address");
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考DisassociateAddress中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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!")
    }
}
```

- 有关 API 的详细信息，请参阅适用DisassociateAddress于 K otlin 的AWS SDK API 参考。

PowerShell

用于 PowerShell

示例 1：此示例取消指定弹性 IP 地址与 VPC 中指定实例的关联。

```
Unregister-EC2Address -AssociationId eipassoc-12345678
```

示例 2：此示例在 EC2-Classic 中取消指定弹性 IP 地址与指定实例的关联。

```
Unregister-EC2Address -PublicIp 203.0.113.17
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[DisassociateAddress](#)中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions."""

    def __init__(self, ec2_resource, elastic_ip=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
 resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param elastic_ip: A Boto3 VpcAddress object. This is a high-level object
 that
                           wraps Elastic IP actions.
        """
        self.ec2_resource = ec2_resource
        self.elastic_ip = elastic_ip

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource)


    def disassociate(self):
```

```
        """
        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.
        """
        if self.elastic_ip is None:
            logger.info("No Elastic IP to disassociate.")
            return

        try:
            self.elastic_ip.association.delete()
        except ClientError as err:
            logger.error(
                "Couldn't disassociate Elastic IP %s from its instance. Here's
why: %s: %s",
                self.elastic_ip.allocation_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- 有关 API 的详细信息，请参阅适用DisassociateAddress于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DisassociateRouteTable`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DisassociateRouteTable`。

CLI

AWS CLI

取消关联路由表

此示例取消指定路由表与指定子网的关联。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 disassociate-route-table --association-id rtbassoc-781d0d1a
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考DisassociateRouteTable中的。

PowerShell

用于 PowerShell

示例 1：此示例删除了路由表和子网之间的指定关联。

```
Unregister-EC2RouteTable -AssociationId rtbassoc-1a2b3c4d
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考DisassociateRouteTable中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

EnableVgwRoutePropagation与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 EnableVgwRoutePropagation。

CLI

AWS CLI

启用路由传播

此示例允许指定的虚拟专用网关将静态路由传播到指定的路由表。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 enable-vgw-route-propagation --route-table-id rtb-22574640 --gateway-id
 vgw-9a4cacf3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考EnableVgwRoutePropagation中的。

PowerShell

用于 PowerShell

示例 1：此示例允许指定的 VGW 自动将路由传播到指定的路由表。

```
Enable-EC2VgwRoutePropagation -RouteTableId rtb-12345678 -GatewayId vgw-1a2b3c4d
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考EnableVgwRoutePropagation中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`EnableVolumeIo`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `EnableVolumeIo`。

CLI

AWS CLI

为卷启用 I/O

此示例在卷上启用 I/O `vol-1234567890abcdef0`。

命令:

```
aws ec2 enable-volume-io --volume-id vol-1234567890abcdef0
```

输出：

```
{
  "Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考EnableVolumeIo中的。

PowerShell

用于 PowerShell

示例 1：如果禁用 I/O 操作，则此示例将为指定卷启用 I/O 操作。

```
Enable-EC2VolumeIO -VolumeId vol-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考EnableVolumeIo中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

EnableVpcClassicLink与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 EnableVpcClassicLink。

CLI

AWS CLI

为其启用 VPC ClassicLink

此示例为启用了 vpc-8888888。 ClassicLink

命令:

```
aws ec2 enable-vpc-classic-link --vpc-id vpc-88888888
```

输出：

```
{
  "Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考EnableVpcClassicLink中的。

PowerShell

用于 PowerShell

示例 1：此示例启用 VPC vpc-0123456b789b0d12f ClassicLink

```
Enable-EC2VpcClassicLink -VpcId vpc-0123456b789b0d12f
```

输出：

```
True
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考EnableVpcClassicLink中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

EnableVpcClassicLinkDnsSupport与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 EnableVpcClassicLinkDnsSupport。

CLI

AWS CLI

为 VPC 启用 ClassicLink DNS 支持

此示例启用了 ClassicLink DNS 支持vpc-88888888。

命令:

```
aws ec2 enable-vpc-classic-link-dns-support --vpc-id vpc-88888888
```

输出：

```
{
  "Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考EnableVpcClassicLinkDnsSupport中的。

PowerShell

用于 PowerShell

示例 1：此示例启用 vpc-0b12d3456a7e8910d 支持 DNS 主机名解析 ClassicLink

```
Enable-EC2VpcClassicLinkDnsSupport -VpcId vpc-0b12d3456a7e8910d -Region eu-west-1
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考EnableVpcClassicLinkDnsSupport中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`GetConsoleOutput`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `GetConsoleOutput`。

CLI

AWS CLI

示例 1：获取控制台输出

以下`get-console-output`示例获取指定 Linux 实例的控制台输出。

```
aws ec2 get-console-output \
    --instance-id i-1234567890abcdef0
```

输出：

```
{
    "InstanceId": "i-1234567890abcdef0",
    "Timestamp": "2013-07-25T21:23:53.000Z",
    "Output": "..."
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的实例控制台输出。

示例 2：获取最新的控制台输出

以下`get-console-output`示例获取指定 Linux 实例的最新控制台输出。

```
aws ec2 get-console-output \
    --instance-id i-1234567890abcdef0 \
    --latest \
    --output text
```

输出：

```
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
```

有关更多信息，请参阅 Amazon EC2 用户指南中的实例控制台输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考GetConsoleOutput中的。

PowerShell

用于 PowerShell

示例 1：此示例获取指定 Linux 实例的控制台输出。控制台输出经过编码。

```
Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456
```

输出：

```
InstanceId          Output
----------          ------
i-0e194d3c47c123637 WyAgICAwLjAwMDAwMF0gQ29tbW...bGU9dHR5UzAgc2Vs
```

示例 2：此示例将编码后的控制台输出存储在变量中，然后对其进行解码。

```
$Output_encoded = (Get-EC2ConsoleOutput -InstanceId i-0e19abcd47c123456).Output
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Output_encoded
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[GetConsoleOutput](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`GetHostReservationPurchasePreview`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `GetHostReservationPurchasePreview`。

CLI

AWS CLI

获取专用主机预留的购买预览

此示例预览了您账户中指定专用主机的指定专用主机预留费用。

命令:

```
aws ec2 get-host-reservation-purchase-preview --offering-id hro-03f707bf363b6b324
 --host-id-set h-013abcd2a00cbd123
```

输出：

```
{
  "TotalHourlyPrice": "1.499",
  "Purchase": [
      {
          "HourlyPrice": "1.499",
          "InstanceFamily": "m4",
          "PaymentOption": "NoUpfront",
          "HostIdSet": [
              "h-013abcd2a00cbd123"
          ],
          "UpfrontPrice": "0.000",
          "Duration": 31536000
      }
  ],
  "TotalUpfrontPrice": "0.000"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[GetHostReservationPurchasePreview](#)中的。

PowerShell

用于 PowerShell

示例 1：此示例预览了与您的专用主机的配置相匹配的预留购买 h-01e23f4cd567890f1

```
Get-EC2HostReservationPurchasePreview -OfferingId hro-0c1f23456789d0ab -HostIdSet
 h-01e23f4cd567890f1
```

输出：

```
CurrencyCode Purchase TotalHourlyPrice TotalUpfrontPrice
------------ -------- ---------------- -----------------
             {}       1.307            0.000
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[GetHostReservationPurchasePreview](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`GetPasswordData`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `GetPasswordData`。

CLI

AWS CLI

要获取加密的密码

此示例获取加密的密码。

命令:

```
aws ec2 get-password-data --instance-id i-1234567890abcdef0
```

输出：

```
{
    "InstanceId": "i-1234567890abcdef0",
    "Timestamp": "2013-08-07T22:18:38.000Z",
    "PasswordData": "gSlJFq+VpcZXqy+iktxMF6NyxQ4qCrT4+gaOuNOenX1MmgXPTj7XEXAMPLE
UQ+YeFfb+L1U4C4AKv652Ux1iRB3CPTYP7WmU3TUnhsuBd+p6LVk7T2lKUml6OXbk6WPW1VYYm/TRPB1
e1DQ7PY4an/DgZT4mwcpRFigzhniQgDDeO1InvSDcwoUTwNs0Y1S8ouri2W4n5GNlriM3Q0AnNVelVz/
53TkDtxbNoU606M1gK9zUWSxqEgwvbV2j8c5rP0WCuaMWSFl4ziDu4bd7q+4RSyi8NUsVWnKZ4aEZffu
DPGzKrF5yLlf3etP2L4ZR6CvG7K1hx7VKOQVN32Dajw=="
}
```

获取解密后的密码

此示例获取解密后的密码。

命令:

```
aws ec2 get-password-data --instance-id  i-1234567890abcdef0 --priv-launch-key C:
\Keys\MyKeyPair.pem
```

输出：

```
{
    "InstanceId": "i-1234567890abcdef0",
    "Timestamp": "2013-08-30T23:18:05.000Z",
    "PasswordData": "&ViJ652e*u"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考GetPasswordData中的。

PowerShell

用于 PowerShell

示例 1：此示例解密 Amazon EC2 分配给指定 Windows 实例管理员账户的密码。指定了 pem 文件后，系统会自动假设-Decrypt 开关的设置。

```
Get-EC2PasswordData -InstanceId i-12345678 -PemFile C:\path\my-key-pair.pem
```

输出：

```
mYZ(PA9?C)Q
```

示例 2：（ PowerShell 仅限 Windows ）检查实例以确定用于启动实例的密钥对的名称，然后尝试在 Visual Studio T AWS oolkit for Visual Studio 的配置存储中查找相应的密钥对数据。如果找到了密钥对数据，则密码将被解密。

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

输出：

```
mYZ(PA9?C)Q
```

示例 3：返回实例的加密密码数据。

```
Get-EC2PasswordData -InstanceId i-12345678
```

输出：

```
iVz3BAK/WAXV.....dqt8WeMA==
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[GetPasswordData](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ImportImage`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ImportImage`。

CLI

AWS CLI

将 VM 映像文件作为 AMI 导入

以下`import-image`示例导入指定的 OVA。

```
aws ec2 import-image \
```

```
  --disk-containers Format=ova,UserBucket="{S3Bucket=my-import-bucket,S3Key=vms/
my-server-vm.ova}"
```

输出：

```
{
    "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"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ImportImage中的。

PowerShell

用于 PowerShell

示例 1：此示例使用等性令牌将单磁盘虚拟机映像从指定的 Amazon S3 存储桶导入到 Amazon EC2。该示例要求存在默认名称为 “vmimport” 的虚拟机导入服务角色，其策略允许 Amazon EC2 访问指定的存储桶，如虚拟机导入先决条件主题中所述。要使用自定义角色，请使用**-RoleName**参数指定角色名称。

```
$container = New-Object Amazon.EC2.Model.ImageDiskContainer
$container.Format="VMDK"
$container.UserBucket = New-Object Amazon.EC2.Model.UserBucket
$container.UserBucket.S3Bucket = "myVirtualMachineImages"
$container.UserBucket.S3Key = "Win_2008_Server_Standard_SP2_64-bit-disk1.vmdk"

$parms = @{
    "ClientToken"="idempotencyToken"
```

```
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
    "LicenseType"="AWS"
}

Import-EC2Image -DiskContainer $container @parms
```

输出：

```
Architecture    :
Description     : Windows 2008 Standard Image
Hypervisor      :
ImageId         :
ImportTaskId    : import-ami-abcdefgh
LicenseType     : AWS
Platform        : Windows
Progress        : 2
SnapshotDetails : {}
Status          : active
StatusMessage   : pending
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ImportImage中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ImportKeyPair`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ImportKeyPair`。

CLI

AWS CLI

导入公钥

首先，使用您选择的工具生成 key pair。例如，使用以下 ssh-keygen 命令：

命令:

```
ssh-keygen -t rsa -C "my-key" -f ~/.ssh/my-key
```

输出：

```
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.
...
```

此示例命令导入指定的公钥。

命令:

```
aws ec2 import-key-pair --key-name "my-key" --public-key-material fileb://~/.ssh/
my-key.pub
```

输出：

```
{
  "KeyName": "my-key",
  "KeyFingerprint": "1f:51:ae:28:bf:89:e9:d8:1f:25:5d:37:2d:7d:b8:ca"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ImportKeyPair中的。

PowerShell

用于 PowerShell

示例 1：此示例将公钥导入 EC2。第一行将公钥文件 (*.pub) 的内容存储在变量中。**$publickey**接下来，该示例将公钥文件的 UTF8 格式转换为 Base64 编码的字符串，并将转换后的字符串存储在变量中。**$pkbase64**在最后一行中，转换后的公钥被导入到 EC2。cmdlet 将密钥指纹和名称作为结果返回。

```
$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
```

输出：

```
KeyFingerprint                                  KeyName
--------------                                  -------
do:d0:15:8f:79:97:12:be:00:fd:df:31:z3:b1:42:z1 Example-user-key
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[ImportKeyPair](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ImportSnapshot`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ImportSnapshot`。

CLI

AWS CLI

导入快照

以下`import-snapshot`示例将指定的磁盘作为快照导入。

```
aws ec2 import-snapshot \
    --description "My server VMDK" \
    --disk-container Format=VMDK,UserBucket={S3Bucket=my-import-bucket,S3Key=vms/
my-server-vm.vmdk}
```

输出：

```
{
    "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"
```

```
        }
    }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ImportSnapshot中的。

PowerShell

用于 PowerShell

示例 1：此示例将格式为 “VMDK” 的虚拟机磁盘映像导入到 Amazon EBS 快照中。该示例需要一个默认名称为 “vmimport” 的虚拟机导入服务角色，其策略允许亚马逊 EC2 访问指定的存储桶，如 http://docs.aws.amazon.com/ AWSEC 2/latestWindowsGuide//VM .html 中的**VM Import Prequisites**主题中所述。ImportPrerequisites要使用自定义角色，请使用**-RoleName**参数指定角色名称。

```
$parms = @{
    "ClientToken"="idempotencyToken"
    "Description"="Disk Image Import"
    "DiskContainer_Description" = "Data disk"
    "DiskContainer_Format" = "VMDK"
    "DiskContainer_S3Bucket" = "myVirtualMachineImages"
    "DiskContainer_S3Key" = "datadiskimage.vmdk"
}

Import-EC2Snapshot @parms
```

输出：

```
Description            ImportTaskId               SnapshotTaskDetail

-----------------      --------------------       ------------------

Disk Image Import      import-snap-abcdefgh
 Amazon.EC2.Model.SnapshotTaskDetail
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ImportSnapshot中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

ModifyCapacityReservation与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 ModifyCapacityReservation。

CLI

AWS CLI

示例 1：更改现有容量预留的实例数量

以下modify-capacity-reservation示例更改了容量预留为其预留容量的实例数量。

```
aws ec2 modify-capacity-reservation \
    --capacity-reservation-id cr-1234abcd56EXAMPLE \
    --instance-count 5
```

输出：

```
{
    "Return": true
}
```

示例 2：更改现有容量预留的结束日期和时间

以下modify-capacity-reservation示例将现有容量预留修改为在指定的日期和时间结束。

```
aws ec2 modify-capacity-reservation \
    --capacity-reservation-id cr-1234abcd56EXAMPLE \
    --end-date-type limited \
    --end-date 2019-08-31T23:59:59Z
```

有关更多信息，请参阅适用于 Linux 实例的 Amazon 弹性计算云用户指南中的修改容量预留。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ModifyCapacityReservation中的。

PowerShell

用于 PowerShell

示例 1：此示例通过将实例计数更改为 1 来修改 CapacityReservationId cr-0c1f2345db6f7cdba

```
Edit-EC2CapacityReservation -CapacityReservationId cr-0c1f2345db6f7cdba -
InstanceCount 1
```

输出：

```
True
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ModifyCapacityReservation中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ModifyHosts`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ModifyHosts`。

CLI

AWS CLI

示例 1：为专用主机启用自动放置

以下`modify-hosts`示例为专用主机启用自动放置，以便其接受与其实例类型配置相匹配的任何非定向实例启动。

```
aws ec2 modify-hosts \
    --host-id h-06c2f189b4EXAMPLE \
    --auto-placement on
```

输出：

```
{
    "Successful": [
        "h-06c2f189b4EXAMPLE"
    ],
    "Unsuccessful": []
}
```

示例 2：为专用主机启用主机恢复

以下modify-hosts示例为指定的专用主机启用主机恢复。

```
aws ec2 modify-hosts \
    --host-id h-06c2f189b4EXAMPLE \
    --host-recovery on
```

输出：

```
{
    "Successful": [
        "h-06c2f189b4EXAMPLE"
    ],
    "Unsuccessful": []
}
```

有关更多信息，请参阅适用于 Linux 实例的 Amazon 弹性计算云用户指南中的修改专用主机的自动放置。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ModifyHosts中的。

PowerShell

用于 PowerShell

示例 1：此示例将专用主机的 AutoPlacement 设置修改为关闭 h-01e23f4cd567890f3

```
Edit-EC2Host -HostId h-03e09f8cd681609f3 -AutoPlacement off
```

输出：

```
Successful            Unsuccessful
----------            ------------
{h-01e23f4cd567890f3} {}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ModifyHosts中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

ModifyIdFormat与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 ModifyIdFormat。

CLI

AWS CLI

为资源启用加长 ID 格式

以下modify-id-format示例为instance资源类型启用了加长 ID 格式。

```
aws ec2 modify-id-format \
    --resource instance \
    --use-long-ids
```

禁用资源的加长 ID 格式

以下modify-id-format示例禁用instance资源类型的加长 ID 格式。

```
aws ec2 modify-id-format \
    --resource instance \
    --no-use-long-ids
```

以下modify-id-format示例为所有处于选择加入期限内的受支持资源类型启用加长 ID 格式。

```
aws ec2 modify-id-format \
    --resource all-current \
    --use-long-ids
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ModifyIdFormat中的。

PowerShell

用于 PowerShell

示例 1：此示例为指定资源类型启用加长 ID 格式。

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

示例 2：此示例禁用指定资源类型的加长 ID 格式。

```
Edit-EC2IdFormat -Resource instance -UseLongId $false
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ModifyIdFormat中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ModifyImageAttribute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ModifyImageAttribute`。

CLI

AWS CLI

示例 1：将 AMI 公开

以下`modify-instance-attribute`示例将指定的 AMI 设为公开。

```
aws ec2 modify-image-attribute \
    --image-id ami-5731123e \
    --launch-permission "Add=[{Group=all}]"
```

此命令不生成任何输出。

示例 2：将 AMI 设为私有

以下`modify-instance-attribute`示例将指定的 AMI 设为私有。

```
aws ec2 modify-image-attribute \
    --image-id ami-5731123e \
    --launch-permission "Remove=[{Group=all}]"
```

此命令不生成任何输出。

示例 3：向 AWS 账户授予启动权限

以下`modify-instance-attribute`示例向指定 AWS 账户授予启动权限。

```
aws ec2 modify-image-attribute \
    --image-id ami-5731123e \
    --launch-permission "Add=[{UserId=123456789012}]"
```

此命令不生成任何输出。

示例 4：移除 AWS 账户的启动权限

以下modify-instance-attribute示例删除了指定 AWS 账户的启动权限。

```
aws ec2 modify-image-attribute \
    --image-id ami-5731123e \
    --launch-permission "Remove=[{UserId=123456789012}]"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ModifyImageAttribute中的。

PowerShell

用于 PowerShell

示例 1：此示例更新了指定 AMI 的描述。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Description "New description"
```

示例 2：此示例将 AMI 设为公开（例如，任何人 AWS 账户 都可以使用）。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserGroup all
```

示例 3：此示例将 AMI 设为私有（例如，只有作为所有者的您才能使用它）。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserGroup all
```

示例 4：此示例向指定的授予启动权限 AWS 账户。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType add -UserId 111122223333
```

示例 5：此示例从指定的中删除启动权限 AWS 账户。

```
Edit-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission -
OperationType remove -UserId 111122223333
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[ModifyImageAttribute](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ModifyInstanceAttribute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ModifyInstanceAttribute`。

CLI

AWS CLI

示例 1：修改实例类型

以下`modify-instance-attribute`示例修改了指定实例的实例类型。该实例必须处于 `stopped` 状态。

```
aws ec2 modify-instance-attribute \
    --instance-id i-1234567890abcdef0 \
    --instance-type "{\"Value\": \"m1.small\"}"
```

此命令不生成任何输出。

示例 2：在实例上启用增强联网

以下`modify-instance-attribute`示例为指定实例启用增强联网。该实例必须处于 `stopped` 状态。

```
aws ec2 modify-instance-attribute \
    --instance-id i-1234567890abcdef0 \
    --sriov-net-support simple
```

此命令不生成任何输出。

示例 3：修改 sourceDestCheck 属性

以下modify-instance-attribute示例将指定实例的sourceDestCheck属性设置为true。该实例必须位于 VPC 中。

```
aws ec2 modify-instance-attribute --instance-id i-1234567890abcdef0 --source-
dest-check "{\"Value\": true}"
```

此命令不生成任何输出。

示例 4：修改根卷的 deleteOnTermination 属性

以下modify-instance-attribute示例将指定 Amazon EBS 支持的实例的根卷的deleteOnTermination属性设置为。false默认情况下，此属性true适用于根卷。

命令:

```
aws ec2 modify-instance-attribute \
  --instance-id i-1234567890abcdef0 \
  --block-device-mappings "[{\"DeviceName\": \"/dev/sda1\",\"Ebs\":
{\"DeleteOnTermination\":false}}]"
```

此命令不生成任何输出。

示例 5：修改附加到实例的用户数据

以下modify-instance-attribute示例将文件内容添加UserData.txt UserData 为指定实例。

原始文件的内容UserData.txt：

```
#!/bin/bash
yum update -y
service httpd start
chkconfig httpd on
```

该文件的内容必须采用 base64 编码。第一个命令将文本文件转换为 base64 并将其另存为新文件。

该命令的 Linux/macOS 版本：

```
base64 UserData.txt > UserData.base64.txt
```

此命令不生成任何输出。

该命令的 Windows 版本：

```
certutil -encode UserData.txt tmp.b64 && findstr /v /c:- tmp.b64 >
 UserData.base64.txt
```

输出：

```
Input Length = 67
Output Length = 152
CertUtil: -encode command completed successfully.
```

现在，你可以在下面的 CLI 命令中引用该文件：

```
aws ec2 modify-instance-attribute \
    --instance-id=i-09b5a14dbca622e76 \
    --attribute userData --value file://UserData.base64.txt
```

此命令不生成任何输出。

有关更多信息，请参阅 EC2 用户指南中的用户数据和 AWS CLI。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ModifyInstanceAttribute中的。

PowerShell

用于 PowerShell

示例 1：此示例修改了指定实例的实例类型。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -InstanceType m3.medium
```

示例 2：此示例通过将 “简单” 指定为单根 I/O 虚拟化 (SR-IOV) 网络支持参数的值，为指定实例启用增强联网，-.. SriovNetSupport

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

示例 3：此示例修改指定实例的安全组。该实例必须位于 VPC 中。必须指定每个安全组的 ID，而不是名称。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -Group @( "sg-12345678",
 "sg-45678901" )
```

示例 4：此示例为指定实例启用 EBS I/O 优化。并非所有实例类型都提供此功能。使用 EBS 优化实例时需要支付额外的使用费。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

示例 5：此示例启用对指定实例的源/目标检查。要让 NAT 实例执行 NAT，该值必须为 “false”。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

示例 6：此示例禁用指定实例的终止功能。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

示例 7：此示例更改了指定的实例，使其在实例启动关闭时终止。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -
InstanceInitiatedShutdownBehavior terminate
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ModifyInstanceAttribute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ModifyInstanceCreditSpecification`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ModifyInstanceCreditSpecification`。

CLI

AWS CLI

修改实例 CPU 使用率的积分选项

此示例将指定区域中指定实例的 CPU 使用率的积分选项修改为 “无限制”。有效的信用选项有 “标准” 和 “无限制”。

命令:

```
aws ec2 modify-instance-credit-specification --instance-credit-specification
 "InstanceId=i-1234567890abcdef0,CpuCredits=unlimited"
```

输出：

```
{
  "SuccessfulInstanceCreditSpecifications": [
    {
      "InstanceId": "i-1234567890abcdef0"
    }
  ],
  "UnsuccessfulInstanceCreditSpecifications": []
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ModifyInstanceCreditSpecification中的。

PowerShell

用于 PowerShell

示例 1：这将启用 T2 无限积分，例如 i-01234567890abcdef。

```
$Credit = New-Object -TypeName
 Amazon.EC2.Model.InstanceCreditSpecificationRequest
$Credit.InstanceId = "i-01234567890abcdef"
$Credit.CpuCredits = "unlimited"
Edit-EC2InstanceCreditSpecification -InstanceCreditSpecification $Credit
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ModifyInstanceCreditSpecification中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ModifyNetworkInterfaceAttribute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ModifyNetworkInterfaceAttribute`。

CLI

AWS CLI

修改网络接口的连接属性

此示例命令修改指定网络接口的attachment属性。

命令:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --
attachment AttachmentId=eni-attach-43348162,DeleteOnTermination=false
```

修改网络接口的描述属性

此示例命令修改指定网络接口的description属性。

命令:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --
description "My description"
```

修改网络接口的 GroupSet 属性

此示例命令修改指定网络接口的groupSet属性。

命令:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --
groups sg-903004f8 sg-1a2b3c4d
```

修改网络接口的 sourceDestCheck 属性

此示例命令修改指定网络接口的sourceDestCheck属性。

命令:

```
aws ec2 modify-network-interface-attribute --network-interface-id eni-686ea200 --
no-source-dest-check
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ModifyNetworkInterfaceAttribute中的。

PowerShell

用于 PowerShell

示例 1：此示例修改了指定的网络接口，以便在终止时删除指定的附件。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

示例 2：此示例修改了指定网络接口的描述。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Description
 "my description"
```

示例 3：此示例修改指定网络接口的安全组。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Groups
 sg-1a2b3c4d
```

示例 4：此示例禁用指定网络接口的源/目标检查。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
SourceDestCheck $false
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[ModifyNetworkInterfaceAttribute](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ModifyReservedInstances`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ModifyReservedInstances`。

CLI

AWS CLI

修改预留实例

此示例命令将预留实例移动到同一地区的另一个可用区。

命令:

```
aws ec2 modify-reserved-instances --reserved-instances-ids b847fa93-
e282-4f55-b59a-1342f5bd7c02 --target-configurations AvailabilityZone=us-
west-1c,Platform=EC2-Classic,InstanceCount=10
```

输出：

```
{
  "ReservedInstancesModificationId": "rimod-d3ed4335-b1d3-4de6-ab31-0f13aaf46687"
}
```

修改预留实例的网络平台

此示例命令将 EC2-Classic 预留实例转换为 EC2-VPC。

命令:

```
aws ec2 modify-reserved-instances --reserved-instances-ids f127bd27-
edb7-44c9-a0eb-0d7e09259af0 --target-configurations AvailabilityZone=us-
west-1c,Platform=EC2-VPC,InstanceCount=5
```

输出：

```
{
  "ReservedInstancesModificationId": "rimod-82fa9020-668f-4fb6-945d-61537009d291"
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的修改您的预留实例。

修改预留实例的实例大小

此示例命令修改在 us-west-1c 中拥有 10 个 m1.small Linux/UNIX 实例的预留实例，这样 8 个 m1.small 实例变成 2 个 m1.large 实例，剩下的 2 个 m1.small 变成 1 个 m1.medium 实例，位于同一可用区。命令:

```
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
```

输出：

```
{
    "ReservedInstancesModificationId": "rimod-acc5f240-080d-4717-
b3e3-1c6b11fa00b6"
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的修改预留实例大小。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ModifyReservedInstances中的。

PowerShell

用于 PowerShell

示例 1：此示例修改了指定预留实例的可用区、实例数量和平台。

```
$config = New-Object Amazon.EC2.Model.ReservedInstancesConfiguration
$config.AvailabilityZone = "us-west-2a"
$config.InstanceCount = 1
$config.Platform = "EC2-VPC"

Edit-EC2ReservedInstance `
-ReservedInstancesId @("FE32132D-70D5-4795-B400-AE435EXAMPLE",
 "0CC556F3-7AB8-4C00-B0E5-98666EXAMPLE") `
-TargetConfiguration $config
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ModifyReservedInstances中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ModifySnapshotAttribute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ModifySnapshotAttribute`。

CLI

AWS CLI

示例 1：修改快照属性

以下modify-snapshot-attribute示例更新了指定快照的createVolumePermission属性，删除了指定用户的卷权限。

```
aws ec2 modify-snapshot-attribute \
    --snapshot-id snap-1234567890abcdef0 \
    --attribute createVolumePermission \
    --operation-type remove \
    --user-ids 123456789012
```

示例 2：公开快照

以下modify-snapshot-attribute示例将指定的快照设为公开。

```
aws ec2 modify-snapshot-attribute \
    --snapshot-id snap-1234567890abcdef0 \
    --attribute createVolumePermission \
    --operation-type add \
    --group-names all
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ModifySnapshotAttribute中的。

PowerShell

用于 PowerShell

示例 1：此示例通过设置指定快照的 CreateVolumePermission 属性将其公开。

```
Edit-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission -OperationType Add -GroupName all
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ModifySnapshotAttribute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

ModifySpotFleetRequest与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 ModifySpotFleetRequest。

CLI

AWS CLI

修改竞价型队列请求

此示例命令更新指定 Spot 队列请求的目标容量。

命令:

```
aws ec2 modify-spot-fleet-request --target-capacity 20 --spot-fleet-request-id
 sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE
```

输出：

```
{
    "Return": true
}
```

此示例命令减少了指定竞价型队列请求的目标容量，但不会因此终止任何竞价型实例。

命令:

```
aws ec2 modify-spot-fleet-request --target-capacity 10 --excess-capacity-
termination-policy NoTermination --spot-fleet-request-ids sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE
```

输出：

```
{
    "Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ModifySpotFleetRequest中的。

PowerShell

用于 PowerShell

示例 1：此示例更新了指定 Spot 队列请求的目标容量。

```
Edit-EC2SpotFleetRequest -SpotFleetRequestId sfr-73fbd2ce-
aa30-494c-8788-1cee4EXAMPLE -TargetCapacity 10
```

输出：

```
True
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[ModifySpotFleetRequest](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ModifySubnetAttribute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ModifySubnetAttribute`。

CLI

AWS CLI

更改子网的公有 IPv4 寻址行为

此示例修改了 subnet-1a2b3c4d，以指定在该子网中启动的所有实例都分配一个公有 IPv4 地址。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 modify-subnet-attribute --subnet-id subnet-1a2b3c4d --map-public-ip-on-
launch
```

更改子网的 IPv6 寻址行为

此示例修改了 subnet-1a2b3c4d，以指定在该子网中启动的所有实例都分配了该子网范围内的 IPv6 地址。

命令:

```
aws ec2 modify-subnet-attribute --subnet-id subnet-1a2b3c4d --assign-ipv6-
address-on-creation
```

有关更多信息，请参阅《AWS 虚拟私有云用户指南》中的 VPC 中的 IP 寻址。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ModifySubnetAttribute中的。

PowerShell

用于 PowerShell

示例 1：此示例为指定子网启用公有 IP 寻址。

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

示例 2：此示例禁用指定子网的公有 IP 寻址。

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $false
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ModifySubnetAttribute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ModifyVolumeAttribute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ModifyVolumeAttribute`。

CLI

AWS CLI

修改体积属性

此示例将 ID 为的卷的`autoEnableIo`属性设置`vol-1234567890abcdef0`为`true`。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 modify-volume-attribute --volume-id vol-1234567890abcdef0 --auto-enable-
io
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ModifyVolumeAttribute中的。

PowerShell

用于 PowerShell

示例 1：此示例修改了指定卷的指定属性。由于数据可能不一致，该卷的 I/O 操作在暂停后会自动恢复。

```
Edit-EC2VolumeAttribute -VolumeId vol-12345678 -AutoEnableIO $true
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ModifyVolumeAttribute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ModifyVpcAttribute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ModifyVpcAttribute`。

CLI

AWS CLI

修改 enableDnsSupport 属性

此示例修改了该`enableDnsSupport`属性。此属性指示 VPC 是否已启用 DNS 解析。如果该属性为 `true`，则 Amazon DNS 服务器会将您实例的 DNS 主机名称解析为相应的 IP 地址，否则不会解析。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 modify-vpc-attribute --vpc-id vpc-a01106c2 --enable-dns-support "{\"Value
\":false}"
```

修改 enableDnsHostnames 属性

此示例修改了该`enableDnsHostnames`属性。此属性表示在 VPC 中启动的实例是否获得 DNS 主机名。如果该属性为 `true`，则 VPC 内的实例可获得 DNS 主机名称，否则将无法获得。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 modify-vpc-attribute --vpc-id vpc-a01106c2 --enable-dns-hostnames
 "{\"Value\":false}"
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ModifyVpcAttribute中的。

PowerShell

用于 PowerShell

示例 1：此示例启用了对指定 VPC 的 DNS 主机名的支持。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $true
```

示例 2：此示例禁用了对指定 VPC 的 DNS 主机名的支持。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsHostnames $false
```

示例 3：此示例支持指定 VPC 的 DNS 解析。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

示例 4：此示例禁用了对指定 VPC 的 DNS 解析支持。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $false
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ModifyVpcAttribute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`MonitorInstances`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `MonitorInstances`。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::MonitorInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    auto dry_run_outcome = ec2Client.MonitorInstances(request);
    if (dry_run_outcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to enable monitoring 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::cerr << "Failed dry run to enable monitoring on instance " <<
                  instanceId << ": " << dry_run_outcome.GetError().GetMessage()
<<
                  std::endl;
        return false;
    }

    request.SetDryRun(false);
    auto 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;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考MonitorInstances中的。

CLI

AWS CLI

启用对实例的详细监控

本示例命令启用对指定实例的详细监控。

命令:

```
aws ec2 monitor-instances --instance-ids i-1234567890abcdef0
```

输出：

```
{
  "InstanceMonitorings": [
      {
          "InstanceId": "i-1234567890abcdef0",
          "Monitoring": {
              "State": "pending"
          }
      }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考MonitorInstances中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { MonitorInstancesCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

// 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.
export const main = async () => {
  const command = new MonitorInstancesCommand({
    InstanceIds: ["INSTANCE_ID"],
  });

  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 (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考MonitorInstances中的。

PowerShell

用于 PowerShell

示例 1：此示例启用了对指定实例的详细监控。

```
Start-EC2InstanceMonitoring -InstanceId i-12345678
```

输出：

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[MonitorInstances](#)中的。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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.
    ENDTRY.
```

- 有关 API 的详细信息，请参阅适用MonitorInstances于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

MoveAddressToVpc与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 MoveAddressToVpc。

CLI

AWS CLI

将地址移至 EC2-VPC

此示例将弹性 IP 地址 54.123.4.56 移至 EC2-VPC 平台。

命令:

```
aws ec2 move-address-to-vpc --public-ip 54.123.4.56
```

输出：

```
{
  "Status": "MoveInProgress"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考MoveAddressToVpc中的。

PowerShell

用于 PowerShell

示例 1：此示例将公有 IP 地址为 12.345.67.89 的 EC2 实例移动到美国东部（弗吉尼亚北部）地区的 EC2-VPC 平台。

```
Move-EC2AddressToVpc -PublicIp 12.345.67.89 -Region us-east-1
```

示例 2：此示例将Get-EC2Instance命令的结果传送到 Move-EC2AddressToVpc cmdlet。该Get-EC2Instance命令获取由实例 ID 指定的实例，然后返回该实例的公有 IP 地址属性。

```
(Get-EC2Instance -Instance i-12345678).Instances.PublicIpAddress | Move-
EC2AddressToVpc
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考MoveAddressToVpc中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`PurchaseHostReservation`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `PurchaseHostReservation`。

CLI

AWS CLI

购买专用主机预留

此示例在您的账户中为指定的专用主机购买指定的专用主机预留服务。

命令:

```
aws ec2 purchase-host-reservation --offering-id hro-03f707bf363b6b324 --host-id-
set h-013abcd2a00cbd123
```

输出：

```
{
  "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"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考PurchaseHostReservation中的。

PowerShell

用于 PowerShell

示例 1：此示例购买的预留配置 hro-0c1f23456789d0ab 的配置与您的专用主机的配置匹配 h-01e23f4cd567890f1

```
New-EC2HostReservation -OfferingId hro-0c1f23456789d0ab HostIdSet
 h-01e23f4cd567890f1
```

输出：

```
ClientToken       :
CurrencyCode      :
Purchase          : {hr-0123f4b5d67bedc89}
TotalHourlyPrice  : 1.307
TotalUpfrontPrice : 0.000
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考PurchaseHostReservation中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

PurchaseScheduledInstances与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 PurchaseScheduledInstances。

CLI

AWS CLI

购买计划实例

此示例购买了计划实例。

命令:

```
aws ec2 purchase-scheduled-instances --purchase-requests file://purchase-
request.json
```

购买请求.json：

```
[
    {
        "PurchaseToken": "eyJ2IjoiMSIsInMiOjEsImMiOi...",
        "InstanceCount": 1
    }
]
```

输出：

```
{
```

```
  "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"
      }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考PurchaseScheduledInstances中的。

PowerShell

用于 PowerShell

示例 1：此示例购买了计划实例。

```
$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOjEsImMiOi..."
New-EC2ScheduledInstancePurchase -PurchaseRequest $request
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考PurchaseScheduledInstances中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

RebootInstances与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 RebootInstances。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 构建和管理弹性服务

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

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

       var response = await _amazonEC2.RebootInstancesAsync(request);
       if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
       {
           Console.WriteLine("Instances successfully rebooted.");
       }
       else
       {
           Console.WriteLine("Could not reboot one or more instances.");
       }
   }
```

替换实例的配置文件、重启并重新启动 Web 服务器。

```
   /// <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)
   {
```

```
        await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
        // Allow time before resetting.
        Thread.Sleep(25000);
        var instanceReady = false;
        var retries = 5;
        while (retries-- > 0 && !instanceReady)
        {
            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(10000);

            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($"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}");
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考[RebootInstances](https://docs.aws.amazon.com/goto/DotNetSDKV3/ec2-2016-11-15/RebootInstances)中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::RebootInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    auto 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);
    auto 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;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考RebootInstances中的。

CLI

AWS CLI

重启 Amazon EC2 实例

本示例将重启指定的实例。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 reboot-instances --instance-ids i-1234567890abcdef5
```

有关更多信息，请参阅《Amazon Elastic Compute Cloud 用户指南》中的“重启实例”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考RebootInstances中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { RebootInstancesCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";
```

```
export const main = async () => {
  const command = new RebootInstancesCommand({
    InstanceIds: ["INSTANCE_ID"],
  });

  try {
    await client.send(command);
    console.log("Instance rebooted successfully.");
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考RebootInstances中的。

PowerShell

用于 PowerShell

示例 1：此示例重新启动指定的实例。

```
Restart-EC2Instance -InstanceId i-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考RebootInstances中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class AutoScaler:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
```

```
    """

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :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
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"
        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"
        self.key_pair_name = f"{resource_prefix}-key-pair"


    def replace_instance_profile(
        self, instance_id, new_instance_profile_name, profile_association_id
    ):
```

```
        """
        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 update.
        :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)
            inst_ready = False
            tries = 0
            while not inst_ready:
                if tries % 6 == 0:
                    self.ec2_client.reboot_instances(InstanceIds=[instance_id])
                    log.info(
                        "Rebooting instance %s and waiting for it to to be
ready.",
                        instance_id,
                    )
                tries += 1
                time.sleep(10)
                response = self.ssm_client.describe_instance_information()
                for info in response["InstanceInformationList"]:
                    if info["InstanceId"] == instance_id:
                        inst_ready = True
            self.ssm_client.send_command(
```

```
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info("Restarted the Python web server on instance %s.",
 instance_id)
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't replace instance profile for association
 {profile_association_id}: {err}"
            )
```

- 有关 API 的详细信息，请参阅适用RebootInstances于 Python 的AWS SDK (Boto3) API 参考。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
async fn reboot_instance(client: &Client, id: &str) -> Result<(), Error> {
    println!("Rebooting instance.");

    client.reboot_instances().instance_ids(id).send().await?;

    client
        .wait_until_instance_stopped()
        .instance_ids(id)
        .wait(Duration::from_secs(60))
        .await?;
    let wait_status_ok = client
        .wait_until_instance_status_ok()
        .instance_ids(id)
        .wait(Duration::from_secs(60))
```

```
        .await;

    match wait_status_ok {
        Ok(_) => println!("Rebooted instance {id}, it is started with status
 OK."),
        Err(err) => return Err(err.into()),
    }

    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用RebootInstances于 Rust 的AWS SDK API 参考。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用RebootInstances于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`RegisterImage`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `RegisterImage`。

CLI

AWS CLI

示例 1：使用清单文件注册 AMI

以下`register-image`示例在 Amazon S3 中使用指定的清单文件注册一个 AMI。

```
aws ec2 register-image \
    --name my-image \
    --image-location my-s3-bucket/myimage/image.manifest.xml
```

输出：

```
{
    "ImageId": "ami-1234567890EXAMPLE"
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的亚马逊机器映像（AMI）。

示例 2：使用根设备的快照注册 AMI

以下register-image示例使用 EBS 根卷的指定快照将 AMI 注册为设备/dev/xvda。块储存设备映射还包括一个空的 100 GiB EBS 卷作为设备。/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}
```

输出：

```
{
    "ImageId": "ami-1a2b3c4d5eEXAMPLE"
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的亚马逊机器映像（AMI）。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考RegisterImage中的。

PowerShell

用于 PowerShell

示例 1：此示例在 Amazon S3 中使用指定的清单文件注册一个 AMI。

```
Register-EC2Image -ImageLocation my-s3-bucket/my-web-server-ami/
image.manifest.xml -Name my-web-server-ami
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考RegisterImage中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

RejectVpcPeeringConnection与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 RejectVpcPeeringConnection。

CLI

AWS CLI

拒绝 VPC 对等连接

此示例拒绝指定的 VPC 对等连接请求。

命令:

```
aws ec2 reject-vpc-peering-connection --vpc-peering-connection-id pcx-1a2b3c4d
```

输出：

```
{
    "Return": true
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考RejectVpcPeeringConnection中的。

PowerShell

用于 PowerShell

示例 1：上面的示例拒绝了请求编号为 pcx-01a2b3ce45fe67eb8 的 VpcPeering 请求

```
Deny-EC2VpcPeeringConnection -VpcPeeringConnectionId pcx-01a2b3ce45fe67eb8
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考RejectVpcPeeringConnection中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

ReleaseAddress与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 ReleaseAddress。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用实例

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    /// <summary>
    /// Release an Elastic IP address.
    /// </summary>
    /// <param name="allocationId">The allocation Id of the Elastic IP address.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> ReleaseAddress(string allocationId)
    {
        var request = new ReleaseAddressRequest
        {
            AllocationId = allocationId
        };

        var response = await _amazonEC2.ReleaseAddressAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考ReleaseAddress中的。

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
###############################################################################
# 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
}
```

本示例中使用的实用程序函数。

```
###############################################################################
# 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
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[ReleaseAddress](#)中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
Aws::EC2::EC2Client ec2(clientConfiguration);

Aws::EC2::Model::ReleaseAddressRequest request;
request.SetAllocationId(allocationID);

auto 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;
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考ReleaseAddress中的。

CLI

AWS CLI

为 EC2-Classic 释放弹性 IP 地址

本示例将释放一个弹性 IP 地址，供 EC2-Classic 中的实例使用。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 release-address --public-ip 198.51.100.0
```

为 EC2-VPC 释放弹性 IP 地址

本示例将释放一个弹性 IP 地址，供 VPC 中的实例使用。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 release-address --allocation-id eipalloc-64d5890a
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ReleaseAddress中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    public static void releaseEC2Address(Ec2Client ec2, String allocId) {
        try {
            ReleaseAddressRequest request = ReleaseAddressRequest.builder()
                    .allocationId(allocId)
                    .build();

            ec2.releaseAddress(request);
            System.out.println("Successfully released Elastic IP address " +
allocId);
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考ReleaseAddress中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { ReleaseAddressCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

export const main = async () => {
  const command = new ReleaseAddressCommand({
    // You can also use PublicIp, but that is for EC2 classic which is being
 retired.
    AllocationId: "ALLOCATION_ID",
  });

  try {
    await client.send(command);
    console.log("Successfully released address.");
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考ReleaseAddress中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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")
    }
}
```

- 有关 API 的详细信息，请参阅适用ReleaseAddress于 K otlin 的AWS SDK API 参考。

PowerShell

用于 PowerShell

示例 1：此示例为 VPC 中的实例释放指定的弹性 IP 地址。

```
Remove-EC2Address -AllocationId eipalloc-12345678 -Force
```

示例 2：此示例为 EC2-Classic 中的实例释放指定的弹性 IP 地址。

```
Remove-EC2Address -PublicIp 198.51.100.2 -Force
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ReleaseAddress中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions."""

    def __init__(self, ec2_resource, elastic_ip=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
 resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param elastic_ip: A Boto3 VpcAddress object. This is a high-level object
 that
                           wraps Elastic IP actions.
        """
        self.ec2_resource = ec2_resource
        self.elastic_ip = elastic_ip

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource)


    def release(self):
        """
        Releases an Elastic IP address. After the Elastic IP address is released,
        it can no longer be used.
        """
        if self.elastic_ip is None:
            logger.info("No Elastic IP to release.")
            return

        try:
            self.elastic_ip.release()
        except ClientError as err:
            logger.error(
                "Couldn't release Elastic IP address %s. Here's why: %s: %s",
                self.elastic_ip.allocation_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- 有关 API 的详细信息，请参阅适用ReleaseAddress于 Python 的AWS SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
# 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)
  return true
rescue StandardError => e
  puts("Error releasing Elastic IP address: #{e.message}")
  return false
end
```

- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考ReleaseAddress中的。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用ReleaseAddress于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ReleaseHosts`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ReleaseHosts`。

CLI

AWS CLI

从您的账户中释放专用主机

从您的账户中释放专用主机。必须先停止或终止主机上的实例，然后才能释放主机。

命令:

```
aws ec2 release-hosts --host-id=h-0029d6e3cacf1b3da
```

输出：

```
{
    "Successful":  [
        "h-0029d6e3cacf1b3da"
         ],
  "Unsuccessful": []

 }
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ReleaseHosts中的。

PowerShell

用于 PowerShell

示例 1：此示例释放给定的主机 ID h-0badafd1dcb2f3456

```
Remove-EC2Host -HostId h-0badafd1dcb2f3456
```

输出：

```
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} {}
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ReleaseHosts中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

ReplaceIamInstanceProfileAssociation与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 ReplaceIamInstanceProfileAssociation。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 构建和管理弹性服务

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
   /// <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)
   {
       await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
               new ReplaceIamInstanceProfileAssociationRequest()
               {
                   AssociationId = associationId,
                   IamInstanceProfile = new IamInstanceProfileSpecification()
```

```
                    {
                        Name = credsProfileName
                    }
                });
        // Allow time before resetting.
        Thread.Sleep(25000);
        var instanceReady = false;
        var retries = 5;
        while (retries-- > 0 && !instanceReady)
        {
            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(10000);

            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($"Sending restart command to instance {instanceId}");
        await _amazonSsm.SendCommandAsync(
            new SendCommandRequest()
            {
                InstanceIds = new List<string>() { instanceId },
                DocumentName = "AWS-RunShellScript",
                Parameters = new Dictionary<string, List<string>>()
                {
                        {"commands", new List<string>() { "cd / && sudo python3
server.py 80" }}
                }
            });
        Console.WriteLine($"Restarted the web server on instance {instanceId}");
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考ReplaceIamInstanceProfileAssociation中的。

CLI

AWS CLI

替换实例的 IAM 实例配置文件

本示例将关联 `iip-assoc-060bae234aac2e7fa` 表示的 IAM 实例配置文件，替换为名为 `AdminRole` 的 IAM 实例配置文件。

```
aws ec2 replace-iam-instance-profile-association \
    --iam-instance-profile Name=AdminRole \
    --association-id iip-assoc-060bae234aac2e7fa
```

输出：

```
{
    "IamInstanceProfileAssociation": {
        "InstanceId": "i-087711ddaf98f9489",
        "State": "associating",
        "AssociationId": "iip-assoc-0b215292fab192820",
        "IamInstanceProfile": {
            "Id": "AIPAJLNLDX3AMYZNWYYAY",
            "Arn": "arn:aws:iam::123456789012:instance-profile/AdminRole"
        }
    }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ReplaceIamInstanceProfileAssociation中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
  await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
    ec2Client.send(
      new ReplaceIamInstanceProfileAssociationCommand({
        AssociationId: state.instanceProfileAssociationId,
        IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
      }),
    ),
  );
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考ReplaceIamInstanceProfileAssociation中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

此示例替换正在运行的实例的实例配置文件，重新启动该实例，并在实例启动后向其发送命令。

```
class AutoScaler:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """
```

```
    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    ):
        """
        :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
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"
        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"
        self.key_pair_name = f"{resource_prefix}-key-pair"


    def replace_instance_profile(
        self, instance_id, new_instance_profile_name, profile_association_id
    ):
        """
```

```
        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 update.
        :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)
            inst_ready = False
            tries = 0
            while not inst_ready:
                if tries % 6 == 0:
                    self.ec2_client.reboot_instances(InstanceIds=[instance_id])
                    log.info(
                        "Rebooting instance %s and waiting for it to to be
ready.",
                        instance_id,
                    )
                tries += 1
                time.sleep(10)
                response = self.ssm_client.describe_instance_information()
                for info in response["InstanceInformationList"]:
                    if info["InstanceId"] == instance_id:
                        inst_ready = True
            self.ssm_client.send_command(
                InstanceIds=[instance_id],
```

```
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info("Restarted the Python web server on instance %s.",
instance_id)
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't replace instance profile for association
{profile_association_id}: {err}"
            )
```

- 有关 API 的详细信息，请参阅适用ReplaceIamInstanceProfileAssociation于 Python 的AWS SDK (Boto3) API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ReplaceNetworkAclAssociation`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ReplaceNetworkAclAssociation`。

CLI

AWS CLI

替换与子网关联的网络 ACL

此示例将指定的网络 ACL 与指定网络 ACL 关联的子网相关联。

命令:

```
aws ec2 replace-network-acl-association --association-id aclassoc-e5b95c8c --
network-acl-id acl-5fb85d36
```

输出：

```
{
    "NewAssociationId": "aclassoc-3999875b"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ReplaceNetworkAclAssociation中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的网络 ACL 与指定网络 ACL 关联的子网相关联。

```
Set-EC2NetworkAclAssociation -NetworkAclId acl-12345678 -AssociationId
 aclassoc-1a2b3c4d
```

输出：

```
aclassoc-87654321
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ReplaceNetworkAclAssociation中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ReplaceNetworkAclEntry`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ReplaceNetworkAclEntry`。

CLI

AWS CLI

替换网络 ACL 条目

此示例替换了指定网络 ACL 的条目。新规则 100 允许通过 UDP 端口 53 (DNS) 从 203.0.113.12/24 进入任何关联子网的入口流量。

命令:

```
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
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ReplaceNetworkAclEntry中的。

PowerShell

用于 PowerShell

示例 1：此示例替换了指定网络 ACL 的指定条目。新规则允许从指定地址到任何关联子网的入站流量。

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ReplaceNetworkAclEntry中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ReplaceRoute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ReplaceRoute`。

CLI

AWS CLI

替换路线

此示例替换了指定路由表中的指定路由。新路由匹配指定的 CIDR，并将流量发送到指定的虚拟专用网关。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 replace-route --route-table-id rtb-22574640 --destination-cidr-block
 10.0.0.0/16 --gateway-id vgw-9a4cacf3
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ReplaceRoute中的。

PowerShell

用于 PowerShell

示例 1：此示例替换指定路由表的指定路由。新路由将指定的流量发送到指定的虚拟专用网关。

```
Set-EC2Route -RouteTableId rtb-1a2b3c4d -DestinationCidrBlock 10.0.0.0/24 -
GatewayId vgw-1a2b3c4d
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ReplaceRoute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ReplaceRouteTableAssociation`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ReplaceRouteTableAssociation`。

CLI

AWS CLI

替换与子网关联的路由表

此示例将指定的路由表与指定路由表关联的子网相关联。

命令:

```
aws ec2 replace-route-table-association --association-id rtbassoc-781d0d1a --
route-table-id rtb-22574640
```

输出：

```
{
    "NewAssociationId": "rtbassoc-3a1f0f58"
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ReplaceRouteTableAssociation中的。

PowerShell

用于 PowerShell

示例 1：此示例将指定的路由表与指定路由表关联的子网相关联。

```
Set-EC2RouteTableAssociation -RouteTableId rtb-1a2b3c4d -AssociationId
 rtbassoc-12345678
```

输出：

```
rtbassoc-87654321
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ReplaceRouteTableAssociation中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ReportInstanceStatus`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ReportInstanceStatus`。

CLI

AWS CLI

报告实例的状态反馈

此示例命令报告指定实例的状态反馈。

命令:

```
aws ec2 report-instance-status --instances i-1234567890abcdef0 --status impaired
 --reason-codes unresponsive
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ReportInstanceStatus中的。

PowerShell

用于 PowerShell

示例 1：此示例报告指定实例的状态反馈。

```
Send-EC2InstanceStatus -Instance i-12345678 -Status impaired -ReasonCode
 unresponsive
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[ReportInstanceStatus](#)中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅[使用 AWS 软件开发工具包创建 Amazon EC2 资源](#)。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`RequestSpotFleet`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `RequestSpotFleet`。

CLI

AWS CLI

请求子网中价格最低的 Spot 队列

此示例命令创建一个 Spot 队列请求，其中包含两个仅因子网而异的启动规格。竞价型队列以最低价格启动指定子网中的实例。如果实例在默认 VPC 中启动，则默认情况下它们会收到一个公有 IP 地址。如果在非默认 VPC 中启动实例，则默认情况下它们不会收到一个公有 IP 地址。

请注意，您不能在竞价型队列请求中指定来自同一可用区的不同子网。

命令:

```
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"
          }
      }
  ]
}
```

输出：

```
{
  "SpotFleetRequestId": "sfr-73fbd2ce-aa30-494c-8788-1cee4EXAMPLE"
}
```

请求可用区内价格最低的 Spot 队列

此示例命令创建具有两个启动规格的 Spot 队列请求，这两个启动规格仅因可用区而异。竞价型队列以最低价格在指定可用区启动实例。如果您的账户仅支持 EC2-VPC，则 Amazon EC2 会在可用区的默认子网中启动竞价型实例。如果您的账户支持 EC2-Classic，则 Amazon EC2 会在可用区启动 EC2-Classic 中的实例。

命令:

```
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"
          }
      }
  ]
}
```

在子网中启动竞价型实例并为其分配公有 IP 地址

此示例命令为在非默认 VPC 中启动的实例分配公有地址。请注意，在指定网络接口时，必须使用该网络接口包括子网 ID 和安全组 ID。

命令:

```
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"
        }
    }
  ]
}
```

使用多元化分配策略请求 Spot 队列

此示例命令创建一个 Spot 队列请求，该请求使用多元化分配策略启动 30 个实例。启动规格因实例类型而异。竞价型队列按启动规格分配实例，因此每种类型有 10 个实例。

命令:

```
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"
        }
    ]
}
```

有关更多信息，请参阅 Amazon 弹性计算云用户指南中的竞价型队列请求。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考RequestSpotFleet中的。

PowerShell

用于 PowerShell

示例 1：此示例在可用区中为指定实例类型创建价格最低的竞价型队列请求。如果您的账户仅支持 EC2-VPC，则竞价型队列会在具有默认子网的价格最低的可用区启动实例。如果您的账户支持 EC2-Classic，则竞价型队列会在价格最低的可用区启动 EC2-Classic 中的实例。请注意，您支付的价格不会超过请求的指定竞价价格。

```
$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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考RequestSpotFleet中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

RequestSpotInstances与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 RequestSpotInstances。

CLI

AWS CLI

请求竞价型实例

此示例命令为指定可用区中的五个实例创建一次性竞价型实例请求。如果您的账户仅支持 EC2-VPC，则 Amazon EC2 会在指定可用区的默认子网中启动实例。如果您的账户支持 EC2-Classic，则 Amazon EC2 会在指定的可用区域中启动 EC2-Classic 中的实例。

命令:

```
aws ec2 request-spot-instances --spot-price "0.03" --instance-count 5 --type
 "one-time" --launch-specification file://specification.json
```

规格.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"
  }
}
```

输出：

```
{
  "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"
        },
        ...
    ]
}
```

此示例命令为指定子网中的五个实例创建一次性竞价型实例请求。Amazon EC2 会在指定子网中启动实例。如果 VPC 是非默认 VPC，则默认情况下实例不会收到公有 IP 地址。

命令:

```
aws ec2 request-spot-instances --spot-price "0.050" --instance-count 5 --type
 "one-time" --launch-specification file://specification.json
```

规格.json：

```
{
  "ImageId": "ami-1a2b3c4d",
  "SecurityGroupIds": [ "sg-1a2b3c4d" ],
```

```
  "InstanceType": "m3.medium",
  "SubnetId": "subnet-1a2b3c4d",
  "IamInstanceProfile": {
      "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
  }
}
```

输出：

```
{
  "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"
        },
        ...
    ]
}
```

此示例为您在非默认 VPC 中启动的竞价型实例分配公有 IP 地址。请注意，在指定网络接口时，必须使用该网络接口包括子网 ID 和安全组 ID。

命令:

```
aws ec2 request-spot-instances --spot-price "0.050" --instance-count 1 --type
 "one-time" --launch-specification file://specification.json
```

规格.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"
  }
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考RequestSpotInstances中的。

PowerShell

用于 PowerShell

示例 1：此示例请求指定子网中的一次性竞价型实例。请注意，必须为包含指定子网的 VPC 创建安全组，并且必须使用网络接口通过 ID 进行指定。指定网络接口时，必须使用该网络接口包括子网 ID。

```
$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
```

输出：

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考RequestSpotInstances中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ResetImageAttribute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ResetImageAttribute`。

CLI

AWS CLI

重置 launchPermission 属性

此示例将指定 AMI 的`launchPermission`属性重置为其默认值。默认情况下，AMI 是私有的。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 reset-image-attribute --image-id ami-5731123e --attribute
 launchPermission
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ResetImageAttribute中的。

PowerShell

用于 PowerShell

示例 1：此示例将 “launchPermission” 属性重置为其默认值。默认情况下，AMI 是私有的。

```
Reset-EC2ImageAttribute -ImageId ami-12345678 -Attribute launchPermission
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ResetImageAttribute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ResetInstanceAttribute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ResetInstanceAttribute`。

CLI

AWS CLI

重置 sourceDestCheck 属性

此示例重置了指定实例的sourceDestCheck属性。该实例必须位于 VPC 中。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --attribute
 sourceDestCheck
```

重置内核属性

此示例重置了指定实例的kernel属性。该实例必须处于 stopped 状态。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --attribute
 kernel
```

重置 ramdisk 属性

此示例重置了指定实例的ramdisk属性。该实例必须处于 stopped 状态。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 reset-instance-attribute --instance-id i-1234567890abcdef0 --attribute
 ramdisk
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ResetInstanceAttribute中的。

PowerShell

用于 PowerShell

示例 1：此示例重置指定实例sriovNetSupport的 “” 属性。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

示例 2：此示例重置指定实例的 “EBSOptimitized” 属性。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute ebsOptimized
```

示例 3：此示例重置指定实例sourceDestCheck的 “” 属性。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sourceDestCheck
```

示例 4：此示例重置指定实例disableApiTermination的 “” 属性。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 disableApiTermination
```

示例 5：此示例重置指定实例的 instanceInitiatedShutdown “行为” 属性。

```
Reset-EC2InstanceAttribute -InstanceId i-12345678 -Attribute
 instanceInitiatedShutdownBehavior
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ResetInstanceAttribute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ResetNetworkInterfaceAttribute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ResetNetworkInterfaceAttribute`。

CLI

AWS CLI

重置网络接口属性

以下`reset-network-interface-attribute`示例将源/目标检查属性的值重置为。`true`

```
aws ec2 reset-network-interface-attribute \
    --network-interface-id eni-686ea200 \
    --source-dest-check
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ResetNetworkInterfaceAttribute中的。

PowerShell

用于 PowerShell

示例 1：此示例重置指定网络接口的源/目标检查。

```
Reset-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -
SourceDestCheck
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ResetNetworkInterfaceAttribute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ResetSnapshotAttribute`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ResetSnapshotAttribute`。

CLI

AWS CLI

重置快照属性

此示例重置了快照`snap-1234567890abcdef0`的创建卷权限。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 reset-snapshot-attribute --snapshot-id snap-1234567890abcdef0 --attribute
 createVolumePermission
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考ResetSnapshotAttribute中的。

PowerShell

用于 PowerShell

示例 1：此示例重置指定快照的指定属性。

```
Reset-EC2SnapshotAttribute -SnapshotId snap-12345678 -Attribute
 CreateVolumePermission
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考ResetSnapshotAttribute中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

RevokeSecurityGroupEgress与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 RevokeSecurityGroupEgress。

CLI

AWS CLI

示例 1：删除允许出站流量到达特定地址范围的规则

以下revoke-security-group-egress示例命令删除了授予对 TCP 端口 80 上指定地址范围的访问权限的规则。

```
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}]
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 用户指南中的安全组。

示例 2：移除允许向特定安全组发送出站流量的规则

以下revoke-security-group-egress示例命令删除了在 TCP 端口 80 上授予对指定安全组的访问权限的规则。

```
aws ec2 revoke-security-group-egress \
    --group-id sg-026c12253ce15eff7 \
    --ip-permissions '[{"IpProtocol": "tcp", "FromPort": 443, "ToPort":
 443,"UserIdGroupPairs": [{"GroupId": "sg-06df23a01ff2df86d"}]}]'
```

此命令不生成任何输出。

有关更多信息，请参阅 Amazon EC2 用户指南中的安全组。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考RevokeSecurityGroupEgress中的。

PowerShell

用于 PowerShell

示例 1：此示例删除了 EC2-VPC 指定安全组的规则。这将撤消对 TCP 端口 80 上指定 IP 地址范围的访问权限。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$ip = @{ IpProtocol="tcp"; FromPort="80"; ToPort="80";
 IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupEgress -GroupId sg-12345678 -IpPermission $ip
```

示例 2：在 PowerShell 版本 2 中，必须使用 New-Object 来创建对象。 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
```

示例 3：此示例撤消对 TCP 端口 80 上指定源安全组的访问权限。

```
$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 } )
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考RevokeSecurityGroupEgress中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

RevokeSecurityGroupIngress与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 RevokeSecurityGroupIngress。

CLI

AWS CLI

示例 1：从安全组中移除规则

以下revoke-security-group-ingress示例从默认 VPC 的指定安全组中删除203.0.113.0/24地址范围的 TCP 端口 22 访问权限。

```
aws ec2 revoke-security-group-ingress \
    --group-name mySecurityGroup
    --protocol tcp \
    --port 22 \
    --cidr 203.0.113.0/24
```

如果成功执行此命令，则不会产生任何输出。

有关更多信息，请参阅 Amazon EC2 用户指南中的安全组。

示例 2：使用 IP 权限集删除规则

以下revoke-security-group-ingress示例使用ip-permissions参数删除允许 ICMP 消息的入站规则Destination Unreachable: Fragmentation Needed and Don't Fragment was Set（类型 3，代码 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}]
```

如果成功执行此命令，则不会产生任何输出。

有关更多信息，请参阅 Amazon EC2 用户指南中的安全组。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考RevokeSecurityGroupIngress中的。

PowerShell

用于 PowerShell

示例 1：此示例从 EC2-VPC 的指定安全组的指定地址范围内撤消对 TCP 端口 22 的访问权限。请注意，您必须使用安全组 ID 而不是安全组名称来识别 EC2-VPC 的安全组。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
 IpRanges="203.0.113.0/24" }
Revoke-EC2SecurityGroupIngress -GroupId sg-12345678 -IpPermission $ip
```

示例 2：在 PowerShell 版本 2 中，必须使用 New-Object 来创建对象。 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
```

示例 3：此示例从 EC2-Classic 的指定安全组的指定地址范围内撤消对 TCP 端口 22 的访问权限。此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
$ip = @{ IpProtocol="tcp"; FromPort="22"; ToPort="22";
 IpRanges="203.0.113.0/24" }

Revoke-EC2SecurityGroupIngress -GroupName "my-security-group" -IpPermission $ip
```

示例 4：在 PowerShell 版本 2 中，必须使用 New-Object 来创建对象。 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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考RevokeSecurityGroupIngress中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

RunInstances与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 RunInstances。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用实例

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    /// <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)
```

```
    {
        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);
        return response.Reservation.Instances[0].InstanceId;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考RunInstances中的。

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
###############################################################################
# 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 getopts 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
```

```
}
```

本示例中使用的实用程序函数。

```
##############################################################################
# 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
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考RunInstances中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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();
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考RunInstances中的。

CLI

AWS CLI

示例 1：将实例启动到默认子网

以下 `run-instances` 示例将类型 `t2.micro` 的单个实例启动到当前区域的默认子网中，并将其与该区域默认 VPC 的默认子网相关联。如果不打算使用 SSH（Linux）或 RDP（Windows）连接到实例，则密钥对是可选的。

```
aws ec2 run-instances \
    --image-id ami-0abcdef1234567890 \
    --instance-type t2.micro \
    --key-name MyKeyPair
```

输出：

```
{
    "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"
}
```

示例 2：将实例启动到非默认子网，并添加一个公有 IP 地址

以下 run-instances 示例为要启动到非默认子网的实例请求一个公有 IP 地址。实例与指定的安全组相关联。

```
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
```

有关 run-instances 的输出示例，请参阅示例 1。

示例 3：启动具有附加卷的实例

以下 run-instances 示例使用 mapping.json 中指定的块设备映射，在启动时对附加卷进行附加。块设备映射可以指定 EBS 卷、实例存储卷，也可以同时指定 EBS 卷和实例存储卷。

```
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
```

mapping.json 的内容。本示例添加了 /dev/sdh，这是一个大小为 100GiB 的空 EBS 卷。

```
[
    {
        "DeviceName": "/dev/sdh",
        "Ebs": {
            "VolumeSize": 100
        }
    }
]
```

mapping.json 的内容。本示例添加了 ephemeral1，作为实例存储卷。

```
[
    {
        "DeviceName": "/dev/sdc",
```

```
        "VirtualName": "ephemeral1"
    }
]
```

有关 run-instances 的输出示例，请参阅示例 1。

有关块设备映射的更多信息，请参阅《Amazon EC2 用户指南》中的块设备映射。

示例 4：启动实例并在创建时添加标签

以下 run-instances 示例向实例中添加了一个键为 webserver、值为 production 的标签。该命令还向创建的任何 EBS 卷 (此示例中为根卷) 应用键为 cost-center、值为 cc123 的标签。

```
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}]'
```

有关 run-instances 的输出示例，请参阅示例 1。

示例 5：启动包含用户数据的实例

以下 run-instances 示例在名为 my_script.txt 的文件中传递用户数据，该文件包含实例的配置脚本。该脚本在启动时运行。

```
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
```

有关 run-instances 的输出示例，请参阅示例 1。

有关实例用户数据的更多信息，请参阅《Amazon EC2 用户指南》中的使用实例用户数据。

示例 6：启动可突增性能实例

以下 run-instances 示例启动带有 unlimited 服务抵扣金选项的 t2.micro 实例。当启动 T2 实例时，如果未指定 --credit-specification，则默认为 standard 服务抵扣金选项。当启动 T3 实例时，默认为 unlimited 服务抵扣金选项。

```
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
```

有关 run-instances 的输出示例，请参阅示例 1。

有关可突增性能实例的更多信息，请参阅《Amazon EC2 用户指南》中的可突增性能实例。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考RunInstances中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2Client;
import software.amazon.awssdk.services.ec2.model.InstanceType;
import software.amazon.awssdk.services.ec2.model.RunInstancesRequest;
import software.amazon.awssdk.services.ec2.model.RunInstancesResponse;
import software.amazon.awssdk.services.ec2.model.Tag;
import software.amazon.awssdk.services.ec2.model.CreateTagsRequest;
import software.amazon.awssdk.services.ec2.model.Ec2Exception;
```

```
/**
 * 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 code example requires an AMI value. You can learn more about this value
 * by reading this documentation topic:
 *
 * https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/AMIs.html
 */
public class CreateInstance {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <name> <amiId>

                Where:
                   name - An instance name value that you can obtain from the AWS
 Console (for example, ami-xxxxxx5c8b987b1a0).\s
                   amiId - An Amazon Machine Image (AMI) value that you can
 obtain from the AWS Console (for example, i-xxxxxx2734106d0ab).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String name = args[0];
        String amiId = args[1];
        Region region = Region.US_EAST_1;
        Ec2Client ec2 = Ec2Client.builder()
            .region(region)
            .build();

        String instanceId = createEC2Instance(ec2, name, amiId);
        System.out.println("The Amazon EC2 Instance ID is " + instanceId);
        ec2.close();
    }
```

```
    public static String createEC2Instance(Ec2Client ec2, String name, String
 amiId) {
        RunInstancesRequest runRequest = RunInstancesRequest.builder()
            .imageId(amiId)
            .instanceType(InstanceType.T1_MICRO)
            .maxCount(1)
            .minCount(1)
            .build();

        // Use a waiter to wait until the instance is running.
        System.out.println("Going to start an EC2 instance using a waiter");
        RunInstancesResponse response = ec2.runInstances(runRequest);
        String instanceIdVal = response.instances().get(0).instanceId();
        ec2.waiter().waitUntilInstanceRunning(r -> r.instanceIds(instanceIdVal));
        Tag tag = Tag.builder()
            .key("Name")
            .value(name)
            .build();

        CreateTagsRequest tagRequest = CreateTagsRequest.builder()
            .resources(instanceIdVal)
            .tags(tag)
            .build();

        try {
            ec2.createTags(tagRequest);
            System.out.printf("Successfully started EC2 Instance %s based on AMI
 %s", instanceIdVal, amiId);
            return instanceIdVal;

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return "";
    }
}
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考[RunInstances](#)中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { RunInstancesCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

// Create a new EC2 instance.
export const main = async () => {
  const command = new RunInstancesCommand({
    // Your key pair name.
    KeyName: "KEY_PAIR_NAME",
    // Your security group.
    SecurityGroupIds: ["SECURITY_GROUP_ID"],
    // An x86_64 compatible image.
    ImageId: "ami-0001a0d1a04bfcc30",
    // An x86_64 compatible free-tier instance type.
    InstanceType: "t1.micro",
    // Ensure only 1 instance launches.
    MinCount: 1,
    MaxCount: 1,
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考RunInstances中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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
    }
}
```

- 有关 API 的详细信息，请参阅适用RunInstances于 K otlin 的AWS SDK API 参考。

PowerShell

用于 PowerShell

示例 1：此示例在 EC2-Classic 或默认 VPC 中启动指定 AMI 的单个实例。

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -InstanceType
 m3.medium -KeyName my-key-pair -SecurityGroup my-security-group
```

示例 2：此示例在 VPC 中启动指定 AMI 的单个实例。

```
New-EC2Instance -ImageId ami-12345678 -MinCount 1 -MaxCount 1 -SubnetId
 subnet-12345678 -InstanceType t2.micro -KeyName my-key-pair -SecurityGroupId
 sg-12345678
```

示例 3：要添加 EBS 卷或实例存储卷，请定义块储存设备映射并将其添加到命令中。此示例添加了一个实例存储卷。

```
$bdm = New-Object Amazon.EC2.Model.BlockDeviceMapping
$bdm.VirtualName = "ephemeral0"
$bdm.DeviceName = "/dev/sdf"

New-EC2Instance -ImageId ami-12345678 -BlockDeviceMapping $bdm ...
```

示例 4：要指定当前的 Windows AMI 之一，请使用Get-EC2ImageByName获取其 AMI ID。此示例从适用于 Windows Server 2016 的当前基础 AMI 启动一个实例。

```
$ami = Get-EC2ImageByName WINDOWS_2016_BASE

New-EC2Instance -ImageId $ami.ImageId ...
```

示例 5：在指定的专用主机环境中启动实例。

```
New-EC2Instance -ImageId ami-1a2b3c4d -InstanceType m4.large -KeyName my-key-pair
 -SecurityGroupId sg-1a2b3c4d  -AvailabilityZone us-west-1a -Tenancy host -HostID
 h-1a2b3c4d5e6f1a2b3
```

示例 6：此请求启动两个实例，并对这些实例应用一个带有 Web 服务器密钥和生产值的标签。该请求还会将密钥为 cost center 且值为 cc123 的标签应用于创建的卷（在本例中为每个实例的根卷）。

```
$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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考RunInstances中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance
 actions."""

    def __init__(self, ec2_resource, instance=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
 resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
```

```
        :param instance: A Boto3 Instance object. This is a high-level object
that
                         wraps instance actions.
        """
        self.ec2_resource = ec2_resource
        self.instance = instance

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource)


    def create(self, image, instance_type, key_pair, security_groups=None):
        """
        Creates a new EC2 instance. The instance starts immediately after
        it is created.

        The instance is created in the default VPC of the current account.

        :param image: A Boto3 Image object that represents an Amazon Machine
Image (AMI)
                      that defines attributes of the instance that is created.
The AMI
                      defines things like the kind of operating system and the
type of
                      storage used by the instance.
        :param instance_type: The type of instance to create, such as 't2.micro'.
                              The instance type defines things like the number of
CPUs and
                              the amount of memory.
        :param key_pair: A Boto3 KeyPair or KeyPairInfo object that represents
the key
                         pair that is used to secure connections to the instance.
        :param security_groups: A list of Boto3 SecurityGroup objects that
represents the
                                security groups that are used to grant access to
the
                                instance. When no security groups are specified,
the
                                default security group of the VPC is used.
        :return: A Boto3 Instance object that represents the newly created
instance.
        """
```

```
        try:
            instance_params = {
                "ImageId": image.id,
                "InstanceType": instance_type,
                "KeyName": key_pair.name,
            }
            if security_groups is not None:
                instance_params["SecurityGroupIds"] = [sg.id for sg in
security_groups]
            self.instance = self.ec2_resource.create_instances(
                **instance_params, MinCount=1, MaxCount=1
            )[0]
            self.instance.wait_until_running()
        except ClientError as err:
            logging.error(
                "Couldn't create instance with image %s, instance type %s, and
key %s. "
                "Here's why: %s: %s",
                image.id,
                instance_type,
                key_pair.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.instance
```

- 有关 API 的详细信息，请参阅适用RunInstances于 Python 的AWS SDK (Boto3) API 参考。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    " Create tags for resource created during instance launch. "
    DATA lt_tagspecifications TYPE /aws1/
cl_ec2tagspecification=>tt_tagspecificationlist.
    DATA ls_tagspecifications LIKE LINE OF lt_tagspecifications.
    ls_tagspecifications =  NEW /aws1/cl_ec2tagspecification(
      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_tagspecifications TO lt_tagspecifications.

    TRY.
        " Create/launch Amazon Elastic Compute Cloud (Amazon EC2) instance. "
        oo_result = lo_ec2->runinstances(                           " oo_result
 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.
```

- 有关 API 的详细信息，请参阅适用RunInstances于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

RunScheduledInstances与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 RunScheduledInstances。

CLI

AWS CLI

启动计划实例

此示例在 VPC 中启动指定的计划实例。

命令:

```
aws ec2 run-scheduled-instances --scheduled-instance-id
 sci-1234-1234-1234-1234-123456789012 --instance-count 1 --launch-specification
 file://launch-specification.json
```

启动规范.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"
  }
}
```

输出：

```
{
  "InstanceIdSet": [
      "i-1234567890abcdef0"
  ]
}
```

此示例在 EC2-Classic 中启动指定的计划实例。

命令:

```
aws ec2 run-scheduled-instances --scheduled-instance-id
 sci-1234-1234-1234-1234-123456789012 --instance-count 1 --launch-specification
 file://launch-specification.json
```

启动规范.json：

```
{
  "ImageId": "ami-12345678",
  "KeyName": "my-key-pair",
  "SecurityGroupIds": ["sg-12345678"],
  "InstanceType": "c4.large",
  "Placement": {
    "AvailabilityZone": "us-west-2b"
  }
  "IamInstanceProfile": {
      "Name": "my-iam-role"
  }
}
```

输出：

```
{
  "InstanceIdSet": [
      "i-1234567890abcdef0"
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考RunScheduledInstances中的。

PowerShell

用于 PowerShell

示例 1：此示例启动指定的计划实例。

```
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
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考RunScheduledInstances中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`StartInstances`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `StartInstances`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用实例

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
/// <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)
{
    var request = new StartInstancesRequest
    {
        InstanceIds = new List<string> { ec2InstanceId },
    };
```

```
        var response = await _amazonEC2.StartInstancesAsync(request);

        if (response.StartingInstances.Count > 0)
        {
            var instances = response.StartingInstances;
            instances.ForEach(i =>
            {
                Console.WriteLine($"Successfully started the EC2 instance with
instance ID: {i.InstanceId}.");
            });
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考StartInstances中的。

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
###############################################################################
# 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 getopts 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
}
```

本示例中使用的实用程序函数。

```
################################################################################
# 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
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考StartInstances中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::StartInstancesRequest start_request;
    start_request.AddInstanceIds(instanceId);
    start_request.SetDryRun(true);

    auto dry_run_outcome = ec2Client.StartInstances(start_request);
    if (dry_run_outcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to start 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 start instance " << instanceId << ": "
                  << dry_run_outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    start_request.SetDryRun(false);
```

```
    auto start_instancesOutcome = ec2Client.StartInstances(start_request);

    if (!start_instancesOutcome.IsSuccess()) {
        std::cout << "Failed to start instance " << instanceId << ": " <<
                  start_instancesOutcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully started instance " << instanceId <<
                  std::endl;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考StartInstances中的。

CLI

AWS CLI

启动 Amazon EC2 实例

本示例启动指定的受 Amazon EBS 支持的实例。

命令:

```
aws ec2 start-instances --instance-ids i-1234567890abcdef0
```

输出：

```
{
    "StartingInstances": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "CurrentState": {
                "Code": 0,
                "Name": "pending"
            },
            "PreviousState": {
                "Code": 80,
                "Name": "stopped"
            }
        }
    ]
```

```
}
```

有关更多信息，请参阅《Amazon Elastic Compute Cloud 用户指南》中的“停止和启动实例”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考StartInstances中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    public static void startInstance(Ec2Client ec2, String instanceId) {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
                .overrideConfiguration(b -> b.maxAttempts(100))
                .client(ec2)
                .build();

        StartInstancesRequest request = StartInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to run.
This will take a few minutes.");
        ec2.startInstances(request);
        DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceRunning(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance " + instanceId);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StartInstances中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { StartInstancesCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

export const main = async () => {
  const command = new StartInstancesCommand({
    // Use DescribeInstancesCommand to find InstanceIds
    InstanceIds: ["INSTANCE_ID"],
  });

  try {
    const { StartingInstances } = await client.send(command);
    const instanceIdList = StartingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Starting instances:");
    console.log(instanceIdList.join("\n"));
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考StartInstances中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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")
    }
}
```

- 有关 API 的详细信息，请参阅适用StartInstances于 K otlin 的AWS SDK API 参考。

PowerShell

用于 PowerShell

示例 1：此示例启动指定的实例。

```
Start-EC2Instance -InstanceId i-12345678
```

输出：

```
CurrentState                        InstanceId    PreviousState
------------                        ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

示例 2：此示例启动指定的实例。

```
@("i-12345678", "i-76543210") | Start-EC2Instance
```

示例 3：此示例启动当前已停止的一组实例。返回的实例对象通过Get-EC2Instance管道传送到。Start-EC2Instance此示例使用的语法需要 PowerShell 版本 3 或更高版本。

```
(Get-EC2Instance -Filter @{ Name="instance-state-name";
 Values="stopped"}).Instances | Start-EC2Instance
```

示例 4：在 PowerShell 版本 2 中，必须使用 New-Object 为 Filter 参数创建过滤器。

```
$filter = New-Object Amazon.EC2.Model.Filter
$filter.Name = "instance-state-name"
$filter.Values = "stopped"

(Get-EC2Instance -Filter $filter).Instances | Start-EC2Instance
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考StartInstances中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance
 actions."""

    def __init__(self, ec2_resource, instance=None):
```

```
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param instance: A Boto3 Instance object. This is a high-level object
that
                            wraps instance actions.
        """
        self.ec2_resource = ec2_resource
        self.instance = instance

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource)


    def start(self):
        """
        Starts an instance and waits for it to be in a running state.

        :return: The response to the start request.
        """
        if self.instance is None:
            logger.info("No instance to start.")
            return

        try:
            response = self.instance.start()
            self.instance.wait_until_running()
        except ClientError as err:
            logger.error(
                "Couldn't start instance %s. Here's why: %s: %s",
                self.instance.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- 有关 API 的详细信息，请参阅适用StartInstances于 Python 的AWS SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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."
  return true
rescue StandardError => e
  puts "Error starting instance: #{e.message}"
  return 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)..."
  unless instance_started?(ec2_client, instance_id)
    puts "Could not start instance."
```

```
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考StartInstances中的。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
async fn start_instance(client: &Client, id: &str) -> Result<(), Error> {
    // start_instance has no unique errors to handle.
    client.start_instances().instance_ids(id).send().await?;

    println!("Waiting for instance to be running");

    let wait_for_running = client
        .wait_until_instance_running()
        .instance_ids(id)
        .wait(Duration::from_secs(60))
        .await;

    match wait_for_running {
        Ok(_) => println!("Instance is running"),
        Err(err) => match err {
            WaiterError::ExceededMaxWait(exceeded) => {
                println!(
                    "Exceeded max time waiting for instance to start. Exceeded
 {}s by {}s.",
                    exceeded.max_wait().as_secs(),
                    (exceeded.elapsed() - exceeded.max_wait()).as_secs()
                );
                return Ok(());
            }
```

```
            _ => return Err(err.into()),
        },
    }

    println!("Started instance.");

    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用StartInstances于 Rust 的AWS SDK API 参考。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用StartInstances于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

StopInstances与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 StopInstances。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用实例

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
/// <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)
{
    // In addition to the list of instance Ids, the
    // request can also include the following properties:
    //     Force      When true, forces the instances to
    //                stop but you must check the integrity
    //                of the file system. Not recommended on
    //                Windows instances.
    //     Hibernate  When true, hibernates the instance if the
    //                instance was enabled for hibernation when
    //                it was launched.
    var request = new StopInstancesRequest
    {
        InstanceIds = new List<string> { ec2InstanceId },
    };

    var response = await _amazonEC2.StopInstancesAsync(request);

    if (response.StoppingInstances.Count > 0)
    {
        var instances = response.StoppingInstances;
        instances.ForEach(i =>
        {
            Console.WriteLine($"Successfully stopped the EC2 Instance " +
                              $"with InstanceID: {i.InstanceId}.");
        });
```

```
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考StopInstances中的。

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
###############################################################################
# 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 getopts 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
}
```

本示例中使用的实用程序函数。

```
###############################################################################
# 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
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[StopInstances](#)中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::StopInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    auto dry_run_outcome = ec2Client.StopInstances(request);
    if (dry_run_outcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to stop 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 stop instance " << instanceId << ": "
                  << dry_run_outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    request.SetDryRun(false);
    auto 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;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考StopInstances中的。

CLI

AWS CLI

示例 1：停止 Amazon EC2 实例

以下 stop-instances 示例将停止指定的受 Amazon EBS 支持的实例。

```
aws ec2 stop-instances \
    --instance-ids i-1234567890abcdef0
```

输出：

```
{
    "StoppingInstances": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "CurrentState": {
                "Code": 64,
                "Name": "stopping"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

有关更多信息，请参阅《Amazon Elastic Compute Cloud 用户指南》中的停止和启动实例。

示例 2：使 Amazon EC2 实例进入休眠状态

以下 stop-instances 示例将让受 Amazon EBS 支持的实例进入休眠，前提是实例已启用休眠并且满足休眠先决条件。实例进入休眠状态后，实例将停止。

```
aws ec2 stop-instances \
    --instance-ids i-1234567890abcdef0 \
    --hibernate
```

输出：

```
{
    "StoppingInstances": [
        {
            "CurrentState": {
                "Code": 64,
                "Name": "stopping"
            },
            "InstanceId": "i-1234567890abcdef0",
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

有关更多信息，请参阅《Amazon Elastic Cloud Compute 用户指南》中的休眠按需 Linux 实例。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考StopInstances中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    public static void stopInstance(Ec2Client ec2, String instanceId) {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
                .overrideConfiguration(b -> b.maxAttempts(100))
                .client(ec2)
                .build();
        StopInstancesRequest request = StopInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();
```

```
        System.out.println("Use an Ec2Waiter to wait for the instance to stop.
This will take a few minutes.");
        ec2.stopInstances(request);
        DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully stopped instance " + instanceId);
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考StopInstances中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { StopInstancesCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

export const main = async () => {
  const command = new StopInstancesCommand({
    // Use DescribeInstancesCommand to find InstanceIds
    InstanceIds: ["INSTANCE_ID"],
  });

  try {
    const { StoppingInstances } = await client.send(command);
    const instanceIdList = StoppingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
```

```
    );
    console.log("Stopping instances:");
    console.log(instanceIdList.join("\n"));
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考StopInstances中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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")
    }
}
```

- 有关 API 的详细信息，请参阅适用StopInstances于 K otlin 的AWS SDK API 参考。

PowerShell

用于 PowerShell

示例 1：此示例停止指定的实例。

```
Stop-EC2Instance -InstanceId i-12345678
```

输出：

```
CurrentState                      InstanceId    PreviousState
------------                      ----------    -------------
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考StopInstances中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance
 actions."""

    def __init__(self, ec2_resource, instance=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
 resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param instance: A Boto3 Instance object. This is a high-level object
 that
                          wraps instance actions.
        """
```

```
        self.ec2_resource = ec2_resource
        self.instance = instance

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource)


    def stop(self):
        """
        Stops an instance and waits for it to be in a stopped state.

        :return: The response to the stop request.
        """
        if self.instance is None:
            logger.info("No instance to stop.")
            return

        try:
            response = self.instance.stop()
            self.instance.wait_until_stopped()
        except ClientError as err:
            logger.error(
                "Couldn't stop instance %s. Here's why: %s: %s",
                self.instance.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response
```

- 有关 API 的详细信息，请参阅适用StopInstances于 Python 的AWS SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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

  ec2_client.stop_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
  puts "Instance stopped."
  return true
rescue StandardError => e
  puts "Error stopping instance: #{e.message}"
  return 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)..."
  unless instance_stopped?(ec2_client, instance_id)
    puts "Could not stop instance."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考StopInstances中的。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
async fn stop_instance(client: &Client, id: &str) -> Result<(), Error> {
    client.stop_instances().instance_ids(id).send().await?;

    println!("Stopping instance...");

    let wait = client
        .wait_until_instance_stopped()
        .instance_ids(id)
        .wait(Duration::from_secs(60))
        .await;

    match wait {
        Ok(_) => {
            println!("Stopped instance.");
        }
        Err(err) => match err {
            WaiterError::ExceededMaxWait(exceeded) => {
                println!(
                    "Exceeded max time waiting for instance to stop. Exceeded {}s
 by {}s",
                    exceeded.max_wait().as_secs(),
                    (exceeded.elapsed() - exceeded.max_wait()).as_secs()
                )
            }
            _ => return Err(err.into()),
        },
    };
    Ok(())
```

```
}
```

- 有关 API 的详细信息，请参阅适用StopInstances于 Rust 的AWS SDK API 参考。

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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.
```

- 有关 API 的详细信息，请参阅适用StopInstances于 S AP 的AWS SDK ABAP API 参考。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`TerminateInstances`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `TerminateInstances`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用实例

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

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

        var response = await _amazonEC2.TerminateInstancesAsync(request);
        return response.TerminatingInstances;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考[TerminateInstances](#)中的。

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
###############################################################################
# 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 getopts 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
}
```

本示例中使用的实用程序函数。

```
##############################################################################
# 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
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考TerminateInstances中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    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;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考TerminateInstances中的。

CLI

AWS CLI

终止 Amazon EC2 实例

本示例将终止指定的实例。

命令:

```
aws ec2 terminate-instances --instance-ids i-1234567890abcdef0
```

输出：

```
{
    "TerminatingInstances": [
        {
            "InstanceId": "i-1234567890abcdef0",
            "CurrentState": {
                "Code": 32,
                "Name": "shutting-down"
            },
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}
```

有关更多信息，请参阅《AWS 命令行界面用户指南》中的“使用 Amazon EC2 实例”。

- 有关 API 的详细信息，请参阅AWS CLI 命令参考TerminateInstances中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    public static void terminateEC2(Ec2Client ec2, String instanceId) {
        try {
            Ec2Waiter ec2Waiter = Ec2Waiter.builder()
                    .overrideConfiguration(b -> b.maxAttempts(100))
                    .client(ec2)
                    .build();

            TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
                    .instanceIds(instanceId)
                    .build();

            System.out.println("Use an Ec2Waiter to wait for the instance to
terminate. This will take a few minutes.");
            ec2.terminateInstances(ti);
            DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
                    .instanceIds(instanceId)
                    .build();

            WaiterResponse<DescribeInstancesResponse> waiterResponse = ec2Waiter
                    .waitUntilInstanceTerminated(instanceRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Successfully started instance " + instanceId);
            System.out.println(instanceId + " is terminated!");

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for Java 2.x API 参考TerminateInstances中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { TerminateInstancesCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

export const main = async () => {
  const command = new TerminateInstancesCommand({
    InstanceIds: ["INSTANCE_ID"],
  });

  try {
    const { TerminatingInstances } = await client.send(command);
    const instanceList = TerminatingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Terminating instances:");
    console.log(instanceList.join("\n"));
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考TerminateInstances中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

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

- 有关 API 的详细信息，请参阅适用TerminateInstances于 K otlin 的AWS SDK API 参考。

PowerShell

用于 PowerShell

示例 1：此示例终止指定的实例（该实例可能正在运行或处于 “已停止” 状态）。在继续操作之前，cmdlet 将提示您进行确认；使用-Force 开关取消该提示。

```
Remove-EC2Instance -InstanceId i-12345678
```

输出：

```
CurrentState                    InstanceId    PreviousState
------------                    ----------    -------------
```

```
Amazon.EC2.Model.InstanceState    i-12345678    Amazon.EC2.Model.InstanceState
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考[TerminateInstances](#)中的。

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance
 actions."""

    def __init__(self, ec2_resource, instance=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
 resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param instance: A Boto3 Instance object. This is a high-level object
 that
                          wraps instance actions.
        """
        self.ec2_resource = ec2_resource
        self.instance = instance

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource)


    def terminate(self):
        """
        Terminates an instance and waits for it to be in a terminated state.
```

```
        """
        if self.instance is None:
            logger.info("No instance to terminate.")
            return

        instance_id = self.instance.id
        try:
            self.instance.terminate()
            self.instance.wait_until_terminated()
            self.instance = None
        except ClientError as err:
            logging.error(
                "Couldn't terminate instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- 有关 API 的详细信息，请参阅适用TerminateInstances于 Python 的AWS SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
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."
  return true
rescue StandardError => e
  puts "Error terminating instance: #{e.message}"
  return 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)..."
  unless instance_terminated?(ec2_client, instance_id)
    puts "Could not terminate instance."
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考TerminateInstances中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`UnassignPrivateIpAddresses`与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 `UnassignPrivateIpAddresses`。

CLI

AWS CLI

从网络接口取消分配辅助私有 IP 地址

此示例取消指定网络接口的指定私有 IP 地址的分配。如果命令成功，则不返回任何输出。

命令:

```
aws ec2 unassign-private-ip-addresses --network-interface-id eni-e5aa89a3 --
private-ip-addresses 10.0.0.82
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考UnassignPrivateIpAddresses中的。

PowerShell

用于 PowerShell

示例 1：此示例取消指定网络接口的指定私有 IP 地址的分配。

```
Unregister-EC2PrivateIpAddress -NetworkInterfaceId eni-1a2b3c4d -PrivateIpAddress
 10.0.0.82
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考UnassignPrivateIpAddresses中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

UnmonitorInstances与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 UnmonitorInstances。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 开始使用实例

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::UnmonitorInstancesRequest unrequest;
    unrequest.AddInstanceIds(instanceId);
    unrequest.SetDryRun(true);

    auto undryRunOutcome = ec2Client.UnmonitorInstances(unrequest);
```

```
    if (undryRunOutcome.IsSuccess()) {
        std::cerr
                << "Failed dry run to disable monitoring on instance. A dry run
should trigger an error."
                <<
                std::endl;
        return false;
    }
    else if (undryRunOutcome.GetError().GetErrorType() !=
             Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to disable monitoring on instance " <<
                  instanceId << ": " << undryRunOutcome.GetError().GetMessage()
<<
                  std::endl;
        return false;
    }

    unrequest.SetDryRun(false);
    auto 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;
    }
```

- 有关 API 的详细信息，请参阅 AWS SDK for C++ API 参考[UnmonitorInstances](#)中的。

CLI

AWS CLI

禁用对实例的详细监控

本示例命令禁用对指定实例的详细监控。

命令:

```
aws ec2 unmonitor-instances --instance-ids i-1234567890abcdef0
```

输出：

```
{
  "InstanceMonitorings": [
      {
          "InstanceId": "i-1234567890abcdef0",
          "Monitoring": {
              "State": "disabling"
          }
      }
  ]
}
```

- 有关 API 的详细信息，请参阅AWS CLI 命令参考[UnmonitorInstances](#)中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
import { UnmonitorInstancesCommand } from "@aws-sdk/client-ec2";

import { client } from "../libs/client.js";

export const main = async () => {
  const command = new UnmonitorInstancesCommand({
    InstanceIds: ["i-09a3dfe7ae00e853f"],
  });

  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 (err) {
    console.error(err);
  }
};
```

- 有关 API 的详细信息，请参阅 AWS SDK for JavaScript API 参考UnmonitorInstances中的。

PowerShell

用于 PowerShell

示例 1：此示例禁用了对指定实例的详细监控。

```
Stop-EC2InstanceMonitoring -InstanceId i-12345678
```

输出：

```
InstanceId    Monitoring
----------    ----------
i-12345678    Amazon.EC2.Model.Monitoring
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考UnmonitorInstances中的。

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS 软件开发工具包的 Amazon EC2 场景

以下代码示例向您展示了如何使用 AWS 软件开发工具包在 Amazon EC2 中实现常见场景。这些场景向您展示了如何通过在 Amazon EC2 中调用多个函数来完成特定任务。每个场景都包含一个指向的链接 GitHub，您可以在其中找到有关如何设置和运行代码的说明。

示例

- 使用 AWS SDK 构建和管理弹性服务
- 使用 AWS 软件开发工具包开始使用 Amazon EC2 实例

使用 AWS SDK 构建和管理弹性服务

以下代码示例展示如何创建可返回书籍、电影和歌曲推荐的负载均衡的 Web 服务。该示例演示服务如何响应故障，以及如何重组服务以提高故障发生时的弹性。

- 使用 Amazon EC2 Auto Scaling 组根据启动模板创建 Amazon Elastic Compute Cloud（Amazon EC2）实例，并将实例数量保持在指定范围内。
- 使用弹性负载均衡处理和分发 HTTP 请求。
- 监控自动扩缩组中实例的运行状况，并仅将请求转发到运行状况良好的实例。
- 在每个 EC2 实例上运行 Python Web 服务器以处理 HTTP 请求。Web 服务器以建议和运行状况检查作为响应。
- 使用 Amazon DynamoDB 表模拟推荐服务。
- 通过更新 AWS Systems Manager 参数来控制 Web 服务器对请求和运行状况检查的响应。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
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($"\nNow, 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($"\nEven 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($"\nWhen 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;
    }
```

创建一个包含自动扩缩和 Amazon EC2 操作的类。

```
/// <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 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)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;

        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)
    {
        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;

    }

    /// <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()
    {
        var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
            new DescribeAvailabilityZonesRequest());
        return zoneResponse.AvailabilityZones.Select(z => z.ZoneName).ToList();
    }

    /// <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()
{
    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];
}

/// <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)
{
    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;
    }

    /// <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 (AmazonClientException)
        {
            Console.WriteLine($"Unable to delete template {templateName}.");
        }
    }

    /// <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)
    {
        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];
    }

    /// <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)
    {
        await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
        // Allow time before resetting.
        Thread.Sleep(25000);
        var instanceReady = false;
        var retries = 5;
        while (retries-- > 0 && !instanceReady)
        {
            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(10000);

            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($"Sending restart command to instance {instanceId}");
        await _amazonSsm.SendCommandAsync(
            new SendCommandRequest()
            {
                InstanceIds = new List<string>() { instanceId },
                DocumentName = "AWS-RunShellScript",
                Parameters = new Dictionary<string, List<string>>()
                {
                        {"commands", new List<string>() { "cd / && sudo python3
server.py 80" }}
                }
            });
        Console.WriteLine($"Restarted the web server on instance {instanceId}");
    }

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

创建一个包含弹性负载均衡操作的类。

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

创建一个使用 DynamoDB 模拟推荐服务的类。

```
/// <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.Write($"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
                        }
```

```
                },
            KeySchema = new List<KeySchemaElement>()
                {
                    new KeySchemaElement()
                    {
                        AttributeName = "MediaType",
                        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.Write("\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.Write(".");
        }
        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");
        }
    }
}
```

创建一个包含 Systems Manager 操作的类。

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

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的以下主题。
  - 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

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
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 wasSuccessul =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            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);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on
for recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto
Scaling instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
```

```
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");

        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the
load balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start
a new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                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 =
HttpClients.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("""
                                Note that it can take a minute or two for the
health check to update
                                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);
    }
}
```

创建一个包含自动扩缩和 Amazon EC2 操作的类。

```
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.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().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.");
    }
}
```

创建一个包含弹性负载均衡操作的类。

```
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 "";
    }
}
```

创建一个使用 DynamoDB 模拟推荐服务的类。

```
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);
    }
}
```

创建一个包含 Systems Manager 操作的类。

```
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);
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的以下主题。
  - 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

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
#!/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 "url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios);
}
```

创建部署所有资源的步骤。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
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 { 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 () => {
    // snippet-start:[javascript.v3.wkflw.resilient.CreateLaunchTemplate]
    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,
        },
      }),
      // snippet-end:[javascript.v3.wkflw.resilient.CreateLaunchTemplate]
    );
  }),
  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) => {
    // snippet-start:[javascript.v3.wkflw.resilient.DescribeVpcs]
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
    // snippet-end:[javascript.v3.wkflw.resilient.DescribeVpcs]
    state.defaultVpc = Vpcs[0].VpcId;
  }),
  new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
  new ScenarioAction("getSubnets", async (state) => {
    // snippet-start:[javascript.v3.wkflw.resilient.DescribeSubnets]
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    // snippet-end:[javascript.v3.wkflw.resilient.DescribeSubnets]
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
```

```
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    // snippet-start:[javascript.v3.wkflw.resilient.CreateTargetGroup]
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    // snippet-end:[javascript.v3.wkflw.resilient.CreateTargetGroup]
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    // snippet-start:[javascript.v3.wkflw.resilient.CreateLoadBalancer]
    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] },
    );
    // snippet-end:[javascript.v3.wkflw.resilient.CreateLoadBalancer]
  }),
  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) => {
    // snippet-start:[javascript.v3.wkflw.resilient.CreateListener]
    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 },
        ],
      }),
    );
    // snippet-end:[javascript.v3.wkflw.resilient.CreateListener]
    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) => {
    // snippet-start:[javascript.v3.wkflw.resilient.AttachTargetGroup]
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
    // snippet-end:[javascript.v3.wkflw.resilient.AttachTargetGroup]
  }),
  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),
      );
    } else {
      return MESSAGES.noIpRules;
    }
  },
),
new ScenarioInput(
  "shouldAddInboundRule",
  /**
   * @param {{ myIpRules: any[] }} state
   */
  (state) => {
    if (state.myIpRules.length > 0) {
      return false;
    } else {
      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);
    } else {
      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,
      );
    }
  }),
];
```

创建运行演示的步骤。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
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) => {
  // snippet-start:[javascript.v3.wkflw.resilient.DescribeTargetGroups]
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.DescribeTargetGroups]

  // snippet-start:[javascript.v3.wkflw.resilient.DescribeTargetHealth]
  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  // snippet-end:[javascript.v3.wkflw.resilient.DescribeTargetHealth]
  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];
      // snippet-start:
[javascript.v3.wkflw.resilient.DescribeIamInstanceProfileAssociations]
      const ec2Client = new EC2Client({});
```

```
      const { IamInstanceProfileAssociations } = await ec2Client.send(
        new DescribeIamInstanceProfileAssociationsCommand({
          Filters: [
            { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
          ],
        }),
      );
      // snippet-end:
[javascript.v3.wkflw.resilient.DescribeIamInstanceProfileAssociations]
      state.instanceProfileAssociationId =
        IamInstanceProfileAssociations[0].AssociationId;
      // snippet-start:
[javascript.v3.wkflw.resilient.ReplaceIamInstanceProfileAssociation]
      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        ec2Client.send(
          new ReplaceIamInstanceProfileAssociationCommand({
            AssociationId: state.instanceProfileAssociationId,
            IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
          }),
        ),
      );
      // snippet-end:
[javascript.v3.wkflw.resilient.ReplaceIamInstanceProfileAssociation]

      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",
    }),
  );
  // snippet-start:[javascript.v3.wkflw.resilient.CreateInstanceProfile]
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
```

```
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  // snippet-end:[javascript.v3.wkflw.resilient.CreateInstanceProfile]
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
  );

  return InstanceProfile;
}
```

创建销毁所有资源的步骤。

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
} 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 { 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 = [
  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(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    } else {
      return MESSAGES.deletedTable.replace("${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,
      );
    } else {
      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);
    } else {
      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,
      );
    } else {
      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);
    } else {
      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,
```

```
      );
    } else {
      return MESSAGES.deletedInstanceRole.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      // snippet-start:[javascript.v3.wkflw.resilient.DeleteInstanceProfile]
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
      // snippet-end:[javascript.v3.wkflw.resilient.DeleteInstanceProfile]
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    } else {
      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(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    } else {
      return MESSAGES.deletedAutoScalingGroup.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      // snippet-start:[javascript.v3.wkflw.resilient.DeleteLaunchTemplate]
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
      // snippet-end:[javascript.v3.wkflw.resilient.DeleteLaunchTemplate]
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    } else {
      return MESSAGES.deletedLaunchTemplate.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
  }),
```

```
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      // snippet-start:[javascript.v3.wkflw.resilient.DeleteLoadBalancer]
      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.");
        }
      });
      // snippet-end:[javascript.v3.wkflw.resilient.DeleteLoadBalancer]
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    } else {
      return MESSAGES.deletedLoadBalancer.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    // snippet-start:[javascript.v3.wkflw.resilient.DeleteTargetGroup]
    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;
  }
  // snippet-end:[javascript.v3.wkflw.resilient.DeleteTargetGroup]
}),
new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
  if (state.deleteLoadBalancerTargetGroupError) {
    console.error(state.deleteLoadBalancerTargetGroupError);
    return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  } else {
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${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);
    } else {
      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);
    } else {
      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");
    } else {
      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(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    } else {
      return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
        "${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(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
    } else {
      return MESSAGES.deletedSsmOnlyPolicy.replace(
        "${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,
      );
    } else {
```

```
      return MESSAGES.deletedSsmOnlyRole.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
  }),
];

/**
 * @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;
    } else {
      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.`);
}
```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的以下主题。
  - 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

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
class Runner:
    def __init__(
        self, resource_path, recommendation, autoscaler, loadbalancer,
 param_helper
    ):
        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

    def deploy(self):
        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"

        print(
            "\nFor this demo, we'll use the AWS SDK for Python (Boto3) 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"
        )
        print(
            "\t* A DynamoDB table that the web service depends on to provide
 book, movie, and song recommendations."
        )
        print(
            "\t* An EC2 launch template that defines EC2 instances that each
 contain a Python web server."
        )
        print(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
 several Availability Zones."
        )
        print(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
 Auto Scaling group to distribute requests."
```

```
        )
        print("-" * 88)
        q.ask("Press Enter when you're ready to start deploying resources.")

        print(
            f"Creating and populating a DynamoDB table named
'{self.recommendation.table_name}'."
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)
        print("-" * 88)

        print(
            f"Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.\n"
            f"This script starts a Python web server defined in the `server.py`
script. The web server\n"
            f"listens to HTTP requests on port 80 and responds to requests to '/'
and to '/healthcheck'.\n"
            f"For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            f"run a web server, such as Apache, with least-privileged
credentials.\n"
        )
        print(
            f"The template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            f"permissions to access the DynamoDB recommendation table and Systems
Manager parameters\n"
            f"that control the flow of the demo.\n"
        )
        self.autoscaler.create_template(startup_script, instance_policy)
        print("-" * 88)

        print(
            f"Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
            f"Availability Zone."
        )
        zones = self.autoscaler.create_group(3)
        print("-" * 88)
        print(
            "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."
        )
        print("-" * 88)
        q.ask("Press Enter when you're ready to continue.")

        print(f"Creating variables that control the flow of the demo.\n")
        self.param_helper.reset()

        print(
            "\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.\n"
        )
        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            [subnet["SubnetId"] for subnet in subnets], target_group
        )
        self.autoscaler.attach_load_balancer_target_group(target_group)
        print(f"Verifying access to the load balancer endpoint...")
        lb_success = self.loadbalancer.verify_load_balancer_endpoint()
        if not lb_success:
            print(
                "Couldn't connect to the load balancer, verifying that the port
is open..."
            )
            current_ip_address = requests.get(
                "http://checkip.amazonaws.com"
            ).text.strip()
            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:
                print(
```

```
                    "For 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 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()
        if lb_success:
            print("Your load balancer is ready. You can access it by browsing to:
\n")
            print(f"\thttp://{self.loadbalancer.endpoint()}\n")
        else:
            print(
                "Couldn'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"
            )
            print(f"\thttp://{self.loadbalancer.endpoint()}\n")
        print("-" * 88)
```

```
        q.ask("Press Enter when you're ready to continue with the demo.")

    def demo_choices(self):
        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:
            print("-" * 88)
            print(
                "\nSee the current state of the service by selecting one of the
following choices:\n"
            )
            choice = q.choose("\nWhich action would you like to take? ", actions)
            print("-" * 88)
            if choice == 0:
                print("Request:\n")
                print(f"GET http://{self.loadbalancer.endpoint()}")
                response = requests.get(f"http://{self.loadbalancer.endpoint()}")
                print("\nResponse:\n")
                print(f"{response.status_code}")
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                print("\nChecking the health of load balancer targets:\n")
                health = self.loadbalancer.check_target_health()
                for target in health:
                    state = target["TargetHealth"]["State"]
                    print(
                        f"\tTarget {target['Target']['Id']} on port
{target['Target']['Port']} is {state}"
                    )
                    if state != "healthy":
                        print(
                            f"\t\t{target['TargetHealth']['Reason']}:
{target['TargetHealth']['Description']}\n"
                        )
                print(
                    f"\nNote that it can take a minute or two for the health
check to update\n"
                    f"after changes are made.\n"
                )
```

```
            elif choice == 2:
                print("\nOkay, let's move on.")
                print("-" * 88)

    def demo(self):
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        print("\nResetting parameters to starting values for demo.\n")
        self.param_helper.reset()

        print(
            "\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."
        )
        print("-" * 88)

        print(
            "At the start, the load balancer endpoint returns recommendations and
reports that all targets are healthy."
        )
        self.demo_choices()

        print(
            f"The web service running on the EC2 instances gets recommendations
by querying a DynamoDB table.\n"
            f"The table name is contained in a Systems Manager parameter named
'{self.param_helper.table}'.\n"
            f"To simulate a failure of the recommendation service, let's set this
parameter to name a non-existent table.\n"
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        print(
            "\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."
        )
        self.demo_choices()

        print(
```

```
            f"Instead of failing when the recommendation service fails, the web
service can return a static response.\n"
            f"While this is not a perfect solution, it presents the customer with
a somewhat better experience than failure.\n"
        )
        self.param_helper.put(self.param_helper.failure_response, "static")
        print(
            f"\nNow, sending a GET request to the load balancer endpoint returns
a static response.\n"
            f"The service still reports as healthy because health checks are
still shallow.\n"
        )
        self.demo_choices()

        print("Let's reinstate the recommendation service.\n")
        self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)
        print(
            "\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"
        )
        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)
        print(
            f"\nReplacing the profile for instance {bad_instance_id} with a
profile that contains\n"
            f"bad credentials...\n"
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
        )
        print(
```

```
            "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"
        )
        self.demo_choices()

        print(
            "\nLet's implement a deep health check. For this demo, a deep health
check tests whether\n"
            "the web service can access the DynamoDB table that it depends on for
recommendations. Note that\n"
            "the deep health check is only for ELB routing and not for Auto
Scaling instance health.\n"
            "This kind of deep health check is not recommended for Auto Scaling
instance health, because it\n"
            "risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.\n"
        )
        print(
            "By implementing deep health checks, the load balancer can detect
when one of the instances is failing\n"
            "and take that instance out of rotation.\n"
        )
        self.param_helper.put(self.param_helper.health_check, "deep")
        print(
            f"\nNow, checking target health indicates that the instance with bad
credentials ({bad_instance_id})\n"
            f"is unhealthy. Note that it might take a minute or two for the load
balancer to detect the unhealthy \n"
            f"instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because\n"
            "the load balancer takes unhealthy instances out of its rotation.\n"
        )
        self.demo_choices()

        print(
            "\nBecause the instances in this demo are controlled by an auto
scaler, the simplest way to fix an unhealthy\n"
            "instance is to terminate it and let the auto scaler start a new
instance to replace it.\n"
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        print(
```

```
            "\nEven while the instance is terminating and the new instance is
starting, sending a GET\n"
            "request to the web service continues to get a successful
recommendation response because\n"
            "the load balancer routes requests to the healthy instances. After
the replacement instance\n"
            "starts and reports as healthy, it is included in the load balancing
rotation.\n"
            "\nNote that terminating and replacing an instance typically takes
several minutes, during which time you\n"
            "can see the changing health check status until the new instance is
running and healthy.\n"
        )
        self.demo_choices()

        print(
            "\nIf the recommendation service fails now, deep health checks mean
all instances report as unhealthy.\n"
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        print(
            "\nWhen all instances are unhealthy, the load balancer continues to
route requests even to\n"
            "unhealthy instances, allowing them to fail open and return a static
response rather than fail\n"
            "closed and report failure to the customer."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self):
        print(
            "This concludes the demo of how to build and manage a resilient
service.\n"
            "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 q.ask("Do you want to clean up all demo resources? (y/n) ",
q.is_yesno):
            self.loadbalancer.delete_load_balancer()
            self.loadbalancer.delete_target_group()
            self.autoscaler.delete_group()
            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,
            )
            self.recommendation.destroy()
        else:
            print(
                "Okay, 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."
            )


def main():
    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()

    print("-" * 88)
    print(
        "Welcome to the demonstration of How to Build and Manage a Resilient
 Service!"
    )
    print("-" * 88)

    prefix = "doc-example-resilience"
    recommendation = RecommendationService.from_client(
        "doc-example-recommendation-service"
    )
```

```
    autoscaler = AutoScaler.from_client(prefix)
    loadbalancer = LoadBalancer.from_client(prefix)
    param_helper = ParameterHelper.from_client(recommendation.table_name)
    runner = Runner(
        args.resource_path, recommendation, autoscaler, loadbalancer,
 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()

    print("-" * 88)
    print("Thanks for watching!")
    print("-" * 88)


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

创建一个包含自动扩缩和 Amazon EC2 操作的类。

```
class AutoScaler:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix,
        inst_type,
        ami_param,
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
```

```
    ):
        """
        :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
        self.launch_template_name = f"{resource_prefix}-template"
        self.group_name = f"{resource_prefix}-group"
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"
        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"
        self.key_pair_name = f"{resource_prefix}-key-pair"


    @classmethod
    def from_client(cls, resource_prefix):
        """
        Creates this class from Boto3 clients.

        :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
        """
        as_client = boto3.client("autoscaling")
        ec2_client = boto3.client("ec2")
        ssm_client = boto3.client("ssm")
        iam_client = boto3.client("iam")
        return cls(
            resource_prefix,
```

```
            "t3.micro",
            "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
            as_client,
            ec2_client,
            ssm_client,
            iam_client,
        )

    def create_instance_profile(
        self, policy_file, policy_name, role_name, profile_name,
aws_managed_policies=()
    ):
        """
        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",
                }
            ],
```

```
        }
        with open(policy_file) as file:
            instance_policy_doc = file.read()

        policy_arn = None
        try:
            pol_response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=instance_policy_doc
            )
            policy_arn = pol_response["Policy"]["Arn"]
            log.info("Created policy with ARN %s.", policy_arn)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                log.info("Policy %s already exists, nothing to do.", policy_name)
                list_pol_response = self.iam_client.list_policies(Scope="Local")
                for pol in list_pol_response["Policies"]:
                    if pol["PolicyName"] == policy_name:
                        policy_arn = pol["Arn"]
                        break
            if policy_arn is None:
                raise AutoScalerError(f"Couldn't create policy {policy_name}:
{err}")

        try:
            self.iam_client.create_role(
                RoleName=role_name,
AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            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("Created role %s and attached policy %s.", role_name,
policy_arn)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                log.info("Role %s already exists, nothing to do.", role_name)
            else:
                raise AutoScalerError(f"Couldn't create role {role_name}: {err}")

        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
                )
            else:
                raise AutoScalerError(
                    f"Couldn't create profile {profile_name} and attach it to
role\n"
                    f"{role_name}: {err}"
                )
        return profile_arn


    def get_instance_profile(self, instance_id):
        """
        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]}]
            )
        except ClientError as err:
```

```
            raise AutoScalerError(
                f"Couldn't get instance profile association for instance
{instance_id}: {err}"
            )
        else:
            return response["IamInstanceProfileAssociations"][0]


    def replace_instance_profile(
        self, instance_id, new_instance_profile_name, profile_association_id
    ):
        """
        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 update.
        :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)
            inst_ready = False
            tries = 0
            while not inst_ready:
                if tries % 6 == 0:
                    self.ec2_client.reboot_instances(InstanceIds=[instance_id])
                    log.info(
```

```
                        "Rebooting instance %s and waiting for it to to be
ready.",
                        instance_id,
                    )
                tries += 1
                time.sleep(10)
                response = self.ssm_client.describe_instance_information()
                for info in response["InstanceInformationList"]:
                    if info["InstanceId"] == instance_id:
                        inst_ready = True
            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info("Restarted the Python web server on instance %s.",
instance_id)
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't replace instance profile for association
{profile_association_id}: {err}"
            )


    def delete_instance_profile(self, profile_name, role_name):
        """
        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:
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )
            else:
                raise AutoScalerError(
                    f"Couldn't delete instance profile {profile_name} or detach "
                    f"policies and delete role {role_name}: {err}"
                )


    def create_key_pair(self, key_pair_name):
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        :return: The newly created key pair.
        """
        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:
            raise AutoScalerError(f"Couldn't create key pair {key_pair_name}:
{err}")


    def delete_key_pair(self):
        """
        Deletes a key pair.

        :param key_pair_name: The name of the key pair to delete.
```

```
        """
        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:
            raise AutoScalerError(
                f"Couldn't delete key pair {self.key_pair_name}: {err}"
            )
        except FileNotFoundError:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
        except PermissionError:
            log.info(
                "Inadequate permissions to delete key pair %s.",
self.key_pair_name
            )
        except Exception as err:
            raise AutoScalerError(
                f"Couldn't delete key pair {self.key_pair_name}: {err}"
            )


    def create_template(self, server_startup_script_file, instance_policy_file):
        """
        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:
```

```
            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,
            )
            with open(server_startup_script_file) as file:
                start_server_script = file.read()
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]
            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(
                "Created launch template %s for AMI %s on %s.",
                self.launch_template_name,
                ami_id,
                self.inst_type,
            )
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.AlreadyExistsException"
            ):
                log.info(
                    "Launch template %s already exists, nothing to do.",
                    self.launch_template_name,
                )
            else:
                raise AutoScalerError(
                    f"Couldn't create launch template
{self.launch_template_name}: {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,
                )
            else:
                raise AutoScalerError(
                    f"Couldn't delete launch template
{self.launch_template_name}: {err}."
                )


    def get_availability_zones(self):
        """
        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"]]
        except ClientError as err:
            raise AutoScalerError(f"Couldn't get availability zones: {err}.")
        else:
```

```
            return zones


    def create_group(self, group_size):
        """
        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.
        """
        zones = []
        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(
                "Created EC2 Auto Scaling group %s with availability zones %s.",
                self.launch_template_name,
                zones,
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "AlreadyExists":
                log.info(
                    "EC2 Auto Scaling group %s already exists, nothing to do.",
                    self.group_name,
                )
            else:
                raise AutoScalerError(
                    f"Couldn't create EC2 Auto Scaling group {self.group_name}:
{err}"
                )
        return zones
```

```
    def get_instances(self):
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: Data about the instances.
        """
        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"]
            ]
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't get instances for Auto Scaling group
{self.group_name}: {err}"
            )
        else:
            return instance_ids


    def terminate_instance(self, instance_id):
        """
        Terminates and instances 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.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id, ShouldDecrementDesiredCapacity=False
            )
            log.info("Terminated instance %s.", instance_id)
        except ClientError as err:
            raise AutoScalerError(f"Couldn't terminate instance {instance_id}:
{err}")

    def attach_load_balancer_target_group(self, lb_target_group):
        """
        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.

        :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:
            raise AutoScalerError(
                f"Couldn't attach load balancer target group
{lb_target_group['TargetGroupName']}\n"
                f"to auto scaling group {self.group_name}"
            )


    def _try_terminate_instance(self, inst_id):
        stopping = False
        log.info(f"Stopping {inst_id}.")
        while not stopping:
            try:
                self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                    InstanceId=inst_id, ShouldDecrementDesiredCapacity=True
                )
                stopping = True
            except ClientError as err:
                if err.response["Error"]["Code"] == "ScalingActivityInProgress":
                    log.info("Scaling activity in progress for %s. Waiting...",
inst_id)
                    time.sleep(10)
                else:
                    raise AutoScalerError(f"Couldn't stop instance {inst_id}:
{err}.")

    def _try_delete_group(self):
```

```
        """
        Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
        the function waits and retries until the group is successfully deleted.
        """
        stopped = False
        while not stopped:
            try:
                self.autoscaling_client.delete_auto_scaling_group(
                    AutoScalingGroupName=self.group_name
                )
                stopped = True
                log.info("Deleted EC2 Auto Scaling group %s.", self.group_name)
            except ClientError as err:
                if (
                    err.response["Error"]["Code"] == "ResourceInUse"
                    or err.response["Error"]["Code"] ==
"ScalingActivityInProgress"
                ):
                    log.info(
                        "Some instances are still running. Waiting for them to
stop..."
                    )
                    time.sleep(10)
                else:
                    raise AutoScalerError(
                        f"Couldn't delete group {self.group_name}: {err}."
                    )

    def delete_group(self):
        """
        Terminates all instances in the group, deletes the EC2 Auto Scaling
group.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=self.group_name, MinSize=0
                )
```

```
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self._try_terminate_instance(inst_id)
                self._try_delete_group()
            else:
                log.info("No groups found named %s, nothing to do.",
 self.group_name)
        except ClientError as err:
            raise AutoScalerError(f"Couldn't delete group {self.group_name}:
 {err}.")


    def get_default_vpc(self):
        """
        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:
            raise AutoScalerError(f"Couldn't get default VPC: {err}")
        else:
            return response["Vpcs"][0]


    def verify_inbound_port(self, vpc, port, ip_address):
        """
        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 specific 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("Found default security group %s.", sec_group["GroupId"])
            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info("Found inbound rule: %s", 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(
                            "The inbound rule does not appear to be open to
either this computer's IP\n"
                            "address of %s, to all IP addresses (0.0.0.0/0), or
to a prefix list ID.",
                            ip_address,
                        )
                    else:
                        break
        except ClientError as err:
            raise AutoScalerError(
                f"Couldn't verify inbound rule for port {port} for VPC
{vpc['VpcId']}: {err}"
            )
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id, port, ip_address):
```

```
        """
        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:
            raise AutoScalerError(
                f"Couldn't authorize ingress to {sec_group_id} on port {port}
from {ip_address}: {err}"
            )


    def get_subnets(self, vpc_id, zones):
        """
        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.
        """
        try:
            response = self.ec2_client.describe_subnets(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
```

```
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )
            subnets = response["Subnets"]
            log.info("Found %s subnets for the specified zones.", len(subnets))
        except ClientError as err:
            raise AutoScalerError(f"Couldn't get subnets: {err}")
        else:
            return subnets
```

创建一个包含弹性负载均衡操作的类。

```
class LoadBalancer:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, target_group_name, load_balancer_name, elb_client):
        """
        :param target_group_name: The name of the target group associated with
 the load balancer.
        :param load_balancer_name: The name of the load balancer.
        :param elb_client: A Boto3 Elastic Load Balancing client.
        """
        self.target_group_name = target_group_name
        self.load_balancer_name = load_balancer_name
        self.elb_client = elb_client
        self._endpoint = None


    @classmethod
    def from_client(cls, resource_prefix):
        """
        Creates this class from a Boto3 client.

        :param resource_prefix: The prefix to give to AWS resources created by
 this class.
        """
        elb_client = boto3.client("elbv2")
        return cls(f"{resource_prefix}-tg", f"{resource_prefix}-lb", elb_client)
```

```
    def endpoint(self):
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        if self._endpoint is None:
            try:
                response = self.elb_client.describe_load_balancers(
                    Names=[self.load_balancer_name]
                )
                self._endpoint = response["LoadBalancers"][0]["DNSName"]
            except ClientError as err:
                raise LoadBalancerError(
                    f"Couldn't get the endpoint for load balancer
{self.load_balancer_name}: {err}"
                )
        return self._endpoint


    def create_target_group(self, protocol, port, vpc_id):
        """
        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.

        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 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=self.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("Created load balancing target group %s.",
self.target_group_name)
        except ClientError as err:
            raise LoadBalancerError(
                f"Couldn't create load balancing target group
{self.target_group_name}: {err}"
            )
        else:
            return target_group


    def delete_target_group(self):
        """
        Deletes the target group.
        """
        done = False
        while not done:
            try:
                response = self.elb_client.describe_target_groups(
                    Names=[self.target_group_name]
                )
                tg_arn = response["TargetGroups"][0]["TargetGroupArn"]
                self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
                log.info(
                    "Deleted load balancing target group %s.",
self.target_group_name
                )
                done = True
            except ClientError as err:
                if err.response["Error"]["Code"] == "TargetGroupNotFound":
                    log.info(
                        "Load balancer target group %s not found, nothing to
do.",
                        self.target_group_name,
                    )
                    done = True
```

```
                elif err.response["Error"]["Code"] == "ResourceInUse":
                    log.info(
                        "Target group not yet released from load balancer,
waiting..."
                    )
                    time.sleep(10)
                else:
                    raise LoadBalancerError(
                        f"Couldn't delete load balancing target group
{self.target_group_name}: {err}"
                    )


    def create_load_balancer(self, subnet_ids, target_group):
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

        :param subnet_ids: A list of subnets to associate with the load balancer.
        :param target_group: An existing target group that is added as a listener
to the
                             load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=self.load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info("Created load balancer %s.", self.load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info("Waiting for load balancer to be available...")
            waiter.wait(Names=[self.load_balancer_name])
            log.info("Load balancer is available!")
            self.elb_client.create_listener(
                LoadBalancerArn=load_balancer["LoadBalancerArn"],
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
```

```
                ],
            )
            log.info(
                "Created listener to forward traffic from load balancer %s to
target group %s.",
                self.load_balancer_name,
                target_group["TargetGroupName"],
            )
        except ClientError as err:
            raise LoadBalancerError(
                f"Failed to create load balancer {self.load_balancer_name}"
                f"and add a listener for target group
{target_group['TargetGroupName']}: {err}"
            )
        else:
            self._endpoint = load_balancer["DNSName"]
            return load_balancer


    def delete_load_balancer(self):
        """
        Deletes a load balancer.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[self.load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", self.load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[self.load_balancer_name])
        except ClientError as err:
            if err.response["Error"]["Code"] == "LoadBalancerNotFound":
                log.info(
                    "Load balancer %s does not exist, nothing to do.",
                    self.load_balancer_name,
                )
            else:
                raise LoadBalancerError(
                    f"Couldn't delete load balancer {self.load_balancer_name}:
{err}"
                )
```

```
   def verify_load_balancer_endpoint(self):
       """
       Verify this computer can successfully send a GET request to the load
balancer endpoint.
       """
       success = False
       retries = 3
       while not success and retries > 0:
           try:
               lb_response = requests.get(f"http://{self.endpoint()}")
               log.info(
                   "Got response %s from load balancer endpoint.",
                   lb_response.status_code,
               )
               if lb_response.status_code == 200:
                   success = True
               else:
                   retries = 0
           except requests.exceptions.ConnectionError:
               log.info(
                   "Got connection error from load balancer endpoint,
retrying..."
               )
               retries -= 1
               time.sleep(10)
       return success

   def check_target_health(self):
       """
       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=[self.target_group_name]
           )
           health_response = self.elb_client.describe_target_health(
               TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
           )
       except ClientError as err:
           raise LoadBalancerError(
```

```
                f"Couldn't check health of {self.target_group_name} targets:
 {err}"
            )
        else:
            return health_response["TargetHealthDescriptions"]
```

创建一个使用 DynamoDB 模拟推荐服务的类。

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
 movies,
    and songs.
    """

    def __init__(self, table_name, dynamodb_client):
        """
        :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

    @classmethod
    def from_client(cls, table_name):
        """
        Creates this class from a Boto3 client.

        :param table_name: The name of the DynamoDB recommendations table.
        """
        ddb_client = boto3.client("dynamodb")
        return cls(table_name, ddb_client)

    def create(self):
        """
        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.

        :return: Data about the newly created table.
        """
        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 do.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file):
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
        """
        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):
        """
        Deletes the recommendations table.
        """
        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}."
                )
```

创建一个包含 Systems Manager 操作的类。

```
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 = "doc-example-resilient-architecture-table"
    failure_response = "doc-example-resilient-architecture-failure-response"
    health_check = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name, ssm_client):
        """
        :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

    @classmethod
    def from_client(cls, table_name):
        ssm_client = boto3.client("ssm")
        return cls(table_name, ssm_client)

    def reset(self):
        """
        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, value):
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting demo parameter %s to '%s'.", name, value)
```

```
        except ClientError as err:
            raise ParameterHelperError(
                f"Couldn't set parameter {name} to {value}: {err}"
            )
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的以下主题。
  - 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

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 AWS 软件开发工具包开始使用 Amazon EC2 实例

以下代码示例演示了如何：

- 创建密钥对和安全组。
- 选择 Amazon 机器映像 (AMI) 和兼容的实例类型，然后创建实例。
- 停止实例，然后再重启。
- 将弹性 IP 地址与您的实例相关联。
- 使用 SSH 连接到您的实例，然后清理资源。

.NET

AWS SDK for .NET

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

在命令提示符中运行场景。

```
/// <summary>
/// Show Amazon Elastic Compute Cloud (Amazon EC2) Basics actions.
/// </summary>
public class EC2Basics
{
    /// <summary>
    /// Perform the actions defined for the Amazon EC2 Basics scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    static async Task Main(string[] args)
    {
```

```
        // Set up dependency injection for Amazon EC2 and Amazon Simple Systems
        // Management Service.
        using var host =
Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonEC2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddTransient<EC2Wrapper>()
                    .AddTransient<SsmWrapper>()
            )
            .Build();

        // Now the client is available for injection.
        var ec2Client = host.Services.GetRequiredService<IAmazonEC2>();
        var ec2Methods = new EC2Wrapper(ec2Client);

        var ssmClient =
host.Services.GetRequiredService<IAmazonSimpleSystemsManagement>();
        var ssmMethods = new SsmWrapper(ssmClient);
        var uiMethods = new UiMethods();

        var uniqueName = Guid.NewGuid().ToString();
        var keyPairName = "mvp-example-key-pair" + uniqueName;
        var groupName = "ec2-scenario-group" + uniqueName;
        var groupDescription = "A security group created for the EC2 Basics
scenario.";

        // Start the scenario.
        uiMethods.DisplayOverview();
        uiMethods.PressEnter();

        // Create the key pair.
        uiMethods.DisplayTitle("Create RSA key pair");
        Console.Write("Let's create an RSA key pair that you can be use to ");
        Console.WriteLine("securely connect to your EC2 instance.");
        var keyPair = await ec2Methods.CreateKeyPair(keyPairName);

        // Save key pair information to a temporary file.
        var tempFileName = ec2Methods.SaveKeyPair(keyPair);

        Console.WriteLine($"Created the key pair: {keyPair.KeyName} and saved it
to: {tempFileName}");
        string? answer;
        do
```

```
        {
            Console.Write("Would you like to list your existing key pairs? ");
            answer = Console.ReadLine();
        } while (answer!.ToLower() != "y" && answer.ToLower() != "n");

        if (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 ec2Methods.DescribeKeyPairs("");
            keyPairs.ForEach(kp =>
            {
                Console.WriteLine($"{kp.KeyName} created at: {kp.CreateTime}
Fingerprint: {kp.KeyFingerprint}");
            });
        }
        uiMethods.PressEnter();

        // Create the security group.
        Console.WriteLine("Let's create a security group to manage access to your
instance.");
        var secGroupId = await ec2Methods.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 ec2Methods.DescribeSecurityGroups(secGroupId);

        Console.WriteLine($"Created security group {groupName} in your default
VPC.");
        secGroups.ForEach(group =>
        {
            ec2Methods.DisplaySecurityGroupInfoAsync(group);
        });
        uiMethods.PressEnter();

        Console.WriteLine("Now we'll authorize the security group we just created
so that it can");
        Console.WriteLine("access the EC2 instances you create.");
        var success = await ec2Methods.AuthorizeSecurityGroupIngress(groupName);
```

```
        secGroups = await ec2Methods.DescribeSecurityGroups(secGroupId);
        Console.WriteLine($"Now let's look at the permissions again.");
        secGroups.ForEach(group =>
        {
            ec2Methods.DisplaySecurityGroupInfoAsync(group);
        });
        uiMethods.PressEnter();

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

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

        var images = await ec2Methods.DescribeImages(imageIds);

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

        int choice;
        bool validNumber = false;

        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
 ec2Methods.DescribeInstanceTypes(selectedImage.Architecture);

        i = 1;
        instanceTypes.ForEach(instanceType =>
        {
            Console.WriteLine($"\t{i++}\t{instanceType.InstanceType}");
```

```
        });

        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");
        var instanceId = await ec2Methods.RunInstances(selectedImage.ImageId,
 selectedInstanceType, keyPairName, secGroupId);
        Console.Write("Waiting for the instance to start.");
        var isRunning = false;
        do
        {
            isRunning = await ec2Methods.WaitForInstanceState(instanceId,
 InstanceStateName.Running);
        } while (!isRunning);

        uiMethods.PressEnter();

        var instance = await ec2Methods.DescribeInstance(instanceId);
        uiMethods.DisplayTitle("New Instance Information");
        ec2Methods.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();

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

        await ec2Methods.StopInstances(instanceId);
        var hasStopped = false;
        do
        {
```

```
            hasStopped = await ec2Methods.WaitForInstanceState(instanceId,
 InstanceStateName.Stopped);
        } while (!hasStopped);

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

        Console.WriteLine("Now let's start it up again.");
        await ec2Methods.StartInstances(instanceId);
        Console.Write("Waiting for instance to start. ");

        isRunning = false;
        do
        {
            isRunning = await ec2Methods.WaitForInstanceState(instanceId,
 InstanceStateName.Running);
        } while (!isRunning);

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

        instance = await ec2Methods.DescribeInstance(instanceId);
        uiMethods.DisplayTitle("New Instance Information");
        ec2Methods.DisplayInstanceInformation(instance);

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

        uiMethods.PressEnter();

        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 ec2Methods.StopInstances(instanceId);
        hasStopped = false;
        do
        {
            hasStopped = await ec2Methods.WaitForInstanceState(instanceId,
 InstanceStateName.Stopped);
        } while (!hasStopped);

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

```
        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 allocationId = await ec2Methods.AllocateAddress();
        Console.WriteLine("Now we will associate the Elastic IP address with our
 instance.");
        var associationId = await ec2Methods.AssociateAddress(allocationId,
 instanceId);

        // Start the instance again.
        Console.WriteLine("Now let's start the instance again.");
        await ec2Methods.StartInstances(instanceId);
        Console.Write("Waiting for instance to start. ");

        isRunning = false;
        do
        {
            isRunning = await ec2Methods.WaitForInstanceState(instanceId,
 InstanceStateName.Running);
        } while (!isRunning);

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

        instance = await ec2Methods.DescribeInstance(instanceId);
        uiMethods.DisplayTitle("Instance information");
        ec2Methods.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();

        await ec2Methods.StopInstances(instanceId);

        hasStopped = false;
        do
        {
            hasStopped = await ec2Methods.WaitForInstanceState(instanceId,
 InstanceStateName.Stopped);
        } while (!hasStopped);
```

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

        Console.WriteLine("Now let's start it up again.");
        await ec2Methods.StartInstances(instanceId);
        Console.Write("Waiting for instance to start. ");

        isRunning = false;
        do
        {
            isRunning = await ec2Methods.WaitForInstanceState(instanceId,
InstanceStateName.Running);
        } while (!isRunning);

        instance = await ec2Methods.DescribeInstance(instanceId);
        uiMethods.DisplayTitle("New Instance Information");
        ec2Methods.DisplayInstanceInformation(instance);
        Console.WriteLine("Note that the IP address did not change this time.");
        uiMethods.PressEnter();

        uiMethods.DisplayTitle("Clean up resources");

        Console.WriteLine("Now let's clean up the resources we created.");

        // Terminate the instance.
        Console.WriteLine("Terminating the instance we created.");
        var stateChange = await ec2Methods.TerminateInstances(instanceId);

        // Wait for the instance state to be terminated.
        var hasTerminated = false;
        do
        {
            hasTerminated = await ec2Methods.WaitForInstanceState(instanceId,
InstanceStateName.Terminated);
        } while (!hasTerminated);

        Console.WriteLine($"\nThe instance {instanceId} has been terminated.");
        Console.WriteLine("Now we can disassociate the Elastic IP address and
release it.");

        // Disassociate the Elastic IP address.
        var disassociated = ec2Methods.DisassociateIp(associationId);

        // Delete the Elastic IP address.
        var released = ec2Methods.ReleaseAddress(allocationId);
```

```
        // Delete the security group.
        Console.WriteLine($"Deleting the Security Group: {groupName}.");
        success = await ec2Methods.DeleteSecurityGroup(secGroupId);
        if (success)
        {
            Console.WriteLine($"Successfully deleted {groupName}.");
        }

        // Delete the RSA key pair.
        Console.WriteLine($"Deleting the key pair: {keyPairName}");
        await ec2Methods.DeleteKeyPair(keyPairName);
        Console.WriteLine("Deleting the temporary file with the key
 information.");
        ec2Methods.DeleteTempFile(tempFileName);
        uiMethods.PressEnter();

        uiMethods.DisplayTitle("EC2 Basics Scenario completed.");
        uiMethods.PressEnter();
    }
}
```

定义一个包装 EC2 操作的类。

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

    public EC2Wrapper(IAmazonEC2 amazonService)
    {
        _amazonEC2 = amazonService;
    }

    /// <summary>
    /// Allocate an Elastic IP address.
    /// </summary>
    /// <returns>The allocation Id of the allocated address.</returns>
    public async Task<string> AllocateAddress()
    {
```

```
        var request = new AllocateAddressRequest();

        var response = await _amazonEC2.AllocateAddressAsync(request);
        return response.AllocationId;
    }

    /// <summary>
    /// Associate an Elastic IP address to an EC2 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)
    {
        var request = new AssociateAddressRequest
        {
            AllocationId = allocationId,
            InstanceId = instanceId
        };

        var response = await _amazonEC2.AssociateAddressAsync(request);
        return response.AssociationId;
    }

    /// <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)
    {
        // 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;
    }

    /// <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.
    /// </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)
    {
        var request = new CreateKeyPairRequest
        {
            KeyName = keyPairName,
        };

        var response = await _amazonEC2.CreateKeyPairAsync(request);

        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
            var kp = response.KeyPair;
```

```
            return kp;
        }
        else
        {
            Console.WriteLine("Could not create key pair.");
            return null;
        }
    }

    /// <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.
    /// </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)
    {
        var response = await _amazonEC2.CreateSecurityGroupAsync(
            new CreateSecurityGroupRequest(groupName, groupDescription));

        return response.GroupId;
    }
```

```
    /// <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 (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)
    {
        var response = await _amazonEC2.DeleteSecurityGroupAsync(new
DeleteSecurityGroupRequest { GroupId = groupId });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <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 existing EC2 images.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task DescribeInstances()
    {
        // List all EC2 instances.
        await GetInstanceDescriptions();

        string tagName = "IncludeInList";
        string tagValue = "Yes";
        await GetInstanceDescriptionsFiltered(tagName, tagValue);
    }

    /// <summary>
    /// Get information for all existing Amazon EC2 instances.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task GetInstanceDescriptions()
    {
        Console.WriteLine("Showing all instances:");
        var paginator = _amazonEC2.Paginators.DescribeInstances(new
DescribeInstancesRequest());

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

```
            {
                foreach (var instance in reservation.Instances)
                {
                    Console.Write($"Instance ID: {instance.InstanceId}");
                    Console.WriteLine($"\tCurrent State: {instance.State.Name}");
                }
            }
        }
    }

    /// <summary>
    /// Get information about EC2 instances filtered by a tag name and value.
    /// </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>Async task.</returns>
    public async Task GetInstanceDescriptionsFiltered(string tagName, string
tagValue)
    {
        // This tag filters the results of the instance list.
        var filters = new List<Filter>
        {
            new Filter
            {
                Name = $"tag:{tagName}",
                Values = new List<string>
                {
                    tagValue,
                },
            },
        };
        var request = new DescribeInstancesRequest
        {
            Filters = filters,
        };

        Console.WriteLine("\nShowing instances with tag: \"IncludeInList\" set to
\"Yes\".");
        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.Write($"Instance ID: {instance.InstanceId} ");
                    Console.WriteLine($"\tCurrent State: {instance.State.Name}");
                }
            }
        }
    }

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

    /// <summary>
    /// Display the instance type information returned by
 DescribeInstanceTypesAsync.
    /// </summary>
    /// <param name="instanceTypes">The list of instance type information.</
param>
    public void DisplayInstanceTypeInfo(List<InstanceTypeInfo> instanceTypes)
    {
        instanceTypes.ForEach(type =>
        {
```

```
            Console.WriteLine($"{type.InstanceType}\t{type.MemoryInfo}");
        });
    }

    /// <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)
    {
        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();
    }


    /// <summary>
    /// Retrieve information for an Amazon EC2 security group.
    /// </summary>
    /// <param name="groupId">The Id of the Amazon EC2 security group.</param>
    /// <returns>A list of security group information.</returns>
    public async Task<List<SecurityGroup>> DescribeSecurityGroups(string groupId)
    {
        var request = new DescribeSecurityGroupsRequest();
        var groupIds = new List<string> { groupId };
        request.GroupIds = groupIds;

        var response = await _amazonEC2.DescribeSecurityGroupsAsync(request);
        return response.SecurityGroups;
    }

    /// <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)
    {
        var response = await _amazonEC2.DisassociateAddressAsync(
            new DisassociateAddressRequest { AssociationId = associationId });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

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

        var response = await _amazonEC2.RebootInstancesAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine("Instances successfully rebooted.");
        }
```

```
        else
        {
            Console.WriteLine("Could not reboot one or more instances.");
        }
    }

    /// <summary>
    /// Release an Elastic IP address.
    /// </summary>
    /// <param name="allocationId">The allocation Id of the Elastic IP address.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> ReleaseAddress(string allocationId)
    {
        var request = new ReleaseAddressRequest
        {
            AllocationId = allocationId
        };

        var response = await _amazonEC2.ReleaseAddressAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <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)
    {
        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);
        return response.Reservation.Instances[0].InstanceId;
    }


    /// <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)
    {
        var request = new StartInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId },
        };

        var response = await _amazonEC2.StartInstancesAsync(request);

        if (response.StartingInstances.Count > 0)
        {
            var instances = response.StartingInstances;
            instances.ForEach(i =>
            {
                Console.WriteLine($"Successfully started the EC2 instance with
instance ID: {i.InstanceId}.");
            });
        }
    }

    /// <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)
    {
        // In addition to the list of instance Ids, the
```

```
        // request can also include the following properties:
        //     Force      When true, forces the instances to
        //                stop but you must check the integrity
        //                of the file system. Not recommended on
        //                Windows instances.
        //     Hibernate  When true, hibernates the instance if the
        //                instance was enabled for hibernation when
        //                it was launched.
        var request = new StopInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId },
        };

        var response = await _amazonEC2.StopInstancesAsync(request);

        if (response.StoppingInstances.Count > 0)
        {
            var instances = response.StoppingInstances;
            instances.ForEach(i =>
            {
                Console.WriteLine($"Successfully stopped the EC2 Instance " +
                                  $"with InstanceID: {i.InstanceId}.");
            });
        }
    }

    /// <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)
    {
        var request = new TerminateInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId }
        };

        var response = await _amazonEC2.TerminateInstancesAsync(request);
        return response.TerminatingInstances;
    }
```

```
    /// <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 running.
        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.Write(". ");
        } while (!hasState);

        return hasState;
    }

}
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API 参考》中的以下主题。
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair

- DeleteSecurityGroup
- DescribeImages
- DescribeInstanceTypes
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DisassociateAddress
- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Bash

AWS CLI 使用 Bash 脚本

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
##############################################################################
# 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

  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

  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 getopts 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 getopts "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
  echo
}
```

此场景中使用的 DynamoDB 函数。

```
###############################################################################
# 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 getopts 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
```

```
}

###############################################################################
# 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 getopts 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 getopts "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 getopts 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
}

###############################################################################
# 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 getopts 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 getopts 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
}

################################################################################
# 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 getopts 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
}

###############################################################################
# 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 getopts 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 getopts 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
}

###############################################################################
# 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 getopts 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 getopts 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
}

################################################################################
# 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 getopts 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 getopts "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 getopts 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 getopts "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
    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
}
```

此场景中使用的实用程序函数。

```
#############################################################################
# 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
}
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances

- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
/**
 * 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 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 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",
 "-");

    public static void main(String[] args) throws InterruptedException {

        final String usage = """

                Usage:
                   <keyName> <fileName> <groupName> <groupDesc> <vpcId>

                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
                   vpcId - A VPC Id value. You can get this value from the AWS
 Management Console.\s
                   myIpAddress - The IP address of your development machine.\s

                """;

        if (args.length != 6) {
            System.out.println(usage);
            System.exit(1);
        }

        String keyName = args[0];
        String fileName = args[1];
        String groupName = args[2];
        String groupDesc = args[3];
        String vpcId = args[4];
        String myIpAddress = args[5];

        Region region = Region.US_WEST_2;
        Ec2Client ec2 = Ec2Client.builder()
                .region(region)
```

```
                .build();

        SsmClient ssmClient = SsmClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon EC2 example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create an RSA key pair and save the private key
material as a .pem file.");
        createKeyPair(ec2, keyName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. List key pairs.");
        describeKeys(ec2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create a security group.");
        String groupId = createSecurityGroup(ec2, groupName, groupDesc, vpcId,
myIpAddress);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Display security group info for the newly created
security group.");
        describeSecurityGroups(ec2, groupId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get a list of Amazon Linux 2 AMIs and selects one
with amzn2 in the name.");
        String instanceId = getParaValues(ssmClient);
        System.out.println("The instance Id is " + instanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Get more information about an amzn2 image.");
        String amiValue = describeImage(ec2, instanceId);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("7. Get a list of instance types.");
        String instanceType = getInstanceTypes(ec2);
        System.out.println("The instance type is " + instanceType);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Create an instance.");
        String newInstanceId = runInstance(ec2, instanceType, keyName, groupName,
amiValue);
        System.out.println("The instance Id is " + newInstanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Display information about the running instance.
");
        String ipAddress = describeEC2Instances(ec2, newInstanceId);
        System.out.println("You can SSH to the instance using this command:");
        System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Stop the instance and use a waiter.");
        stopInstance(ec2, newInstanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Start the instance and use a waiter.");
        startInstance(ec2, newInstanceId);
        ipAddress = describeEC2Instances(ec2, newInstanceId);
        System.out.println("You can SSH to the instance using this command:");
        System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Allocate an Elastic IP address and associate it
with the instance.");
        String allocationId = allocateAddress(ec2);
        System.out.println("The allocation Id value is " + allocationId);
        String associationId = associateAddress(ec2, newInstanceId,
allocationId);
        System.out.println("The associate Id value is " + associationId);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("13. Describe the instance again.");
        ipAddress = describeEC2Instances(ec2, newInstanceId);
        System.out.println("You can SSH to the instance using this command:");
        System.out.println("ssh -i " + fileName + "ec2-user@" + ipAddress);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Disassociate and release the Elastic IP
address.");
        disassociateAddress(ec2, associationId);
        releaseEC2Address(ec2, allocationId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Terminate the instance and use a waiter.");
        terminateEC2(ec2, newInstanceId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete the security group.");
        deleteEC2SecGroup(ec2, groupId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Delete the key.");
        deleteKeys(ec2, keyName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("You successfully completed the Amazon EC2
scenario.");
        System.out.println(DASHES);
        ec2.close();
    }

    public static void deleteEC2SecGroup(Ec2Client ec2, String groupId) {
        try {
            DeleteSecurityGroupRequest request =
DeleteSecurityGroupRequest.builder()
                    .groupId(groupId)
                    .build();
```

```
            ec2.deleteSecurityGroup(request);
            System.out.println("Successfully deleted security group with Id " +
groupId);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void terminateEC2(Ec2Client ec2, String instanceId) {
        try {
            Ec2Waiter ec2Waiter = Ec2Waiter.builder()
                    .overrideConfiguration(b -> b.maxAttempts(100))
                    .client(ec2)
                    .build();

            TerminateInstancesRequest ti = TerminateInstancesRequest.builder()
                    .instanceIds(instanceId)
                    .build();

            System.out.println("Use an Ec2Waiter to wait for the instance to
terminate. This will take a few minutes.");
            ec2.terminateInstances(ti);
            DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
                    .instanceIds(instanceId)
                    .build();

            WaiterResponse<DescribeInstancesResponse> waiterResponse = ec2Waiter
                    .waitUntilInstanceTerminated(instanceRequest);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Successfully started instance " + instanceId);
            System.out.println(instanceId + " is terminated!");

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteKeys(Ec2Client ec2, String keyPair) {
        try {
            DeleteKeyPairRequest request = DeleteKeyPairRequest.builder()
```

```
                    .keyName(keyPair)
                    .build();

            ec2.deleteKeyPair(request);
            System.out.println("Successfully deleted key pair named " + keyPair);

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void releaseEC2Address(Ec2Client ec2, String allocId) {
        try {
            ReleaseAddressRequest request = ReleaseAddressRequest.builder()
                    .allocationId(allocId)
                    .build();

            ec2.releaseAddress(request);
            System.out.println("Successfully released Elastic IP address " +
allocId);
        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void disassociateAddress(Ec2Client ec2, String associationId) {
        try {
            DisassociateAddressRequest addressRequest =
DisassociateAddressRequest.builder()
                    .associationId(associationId)
                    .build();

            ec2.disassociateAddress(addressRequest);
            System.out.println("You successfully disassociated the address!");

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
   public static String associateAddress(Ec2Client ec2, String instanceId,
String allocationId) {
       try {
           AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
                   .instanceId(instanceId)
                   .allocationId(allocationId)
                   .build();

           AssociateAddressResponse associateResponse =
ec2.associateAddress(associateRequest);
           return associateResponse.associationId();

       } catch (Ec2Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }

   public static String allocateAddress(Ec2Client ec2) {
       try {
           AllocateAddressRequest allocateRequest =
AllocateAddressRequest.builder()
                   .domain(DomainType.VPC)
                   .build();

           AllocateAddressResponse allocateResponse =
ec2.allocateAddress(allocateRequest);
           return allocateResponse.allocationId();

       } catch (Ec2Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return "";
   }

   public static void startInstance(Ec2Client ec2, String instanceId) {
       Ec2Waiter ec2Waiter = Ec2Waiter.builder()
               .overrideConfiguration(b -> b.maxAttempts(100))
               .client(ec2)
               .build();
```

```
        StartInstancesRequest request = StartInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to run.
This will take a few minutes.");
        ec2.startInstances(request);
        DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceRunning(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully started instance " + instanceId);
    }

    public static void stopInstance(Ec2Client ec2, String instanceId) {
        Ec2Waiter ec2Waiter = Ec2Waiter.builder()
                .overrideConfiguration(b -> b.maxAttempts(100))
                .client(ec2)
                .build();
        StopInstancesRequest request = StopInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        System.out.println("Use an Ec2Waiter to wait for the instance to stop.
This will take a few minutes.");
        ec2.stopInstances(request);
        DescribeInstancesRequest instanceRequest =
DescribeInstancesRequest.builder()
                .instanceIds(instanceId)
                .build();

        WaiterResponse<DescribeInstancesResponse> waiterResponse =
ec2Waiter.waitUntilInstanceStopped(instanceRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Successfully stopped instance " + instanceId);
    }

    public static String describeEC2Instances(Ec2Client ec2, String
newInstanceId) {
        try {
```

```
            String pubAddress = "";
            boolean isRunning = false;
            DescribeInstancesRequest request = DescribeInstancesRequest.builder()
                    .instanceIds(newInstanceId)
                    .build();

            while (!isRunning) {
                DescribeInstancesResponse response =
ec2.describeInstances(request);
                String state =
response.reservations().get(0).instances().get(0).state().name().name();
                if (state.compareTo("RUNNING") == 0) {
                    System.out.println("Image id is " +
response.reservations().get(0).instances().get(0).imageId());
                    System.out.println(
                            "Instance type is " +
response.reservations().get(0).instances().get(0).instanceType());
                    System.out.println(
                            "Instance state is " +
response.reservations().get(0).instances().get(0).state().name());
                    pubAddress =
response.reservations().get(0).instances().get(0).publicIpAddress();
                    System.out.println("Instance address is " + pubAddress);
                    isRunning = true;
                }
            }
            return pubAddress;
        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    public static String runInstance(Ec2Client ec2, String instanceType, String
keyName, String groupName,
            String amiId) {
        try {
            RunInstancesRequest runRequest = RunInstancesRequest.builder()
                    .instanceType(instanceType)
                    .keyName(keyName)
                    .securityGroups(groupName)
                    .maxCount(1)
                    .minCount(1)
```

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

            System.out.println("Going to start an EC2 instance using a waiter");
            RunInstancesResponse response = ec2.runInstances(runRequest);
            String instanceIdVal = response.instances().get(0).instanceId();
            ec2.waiter().waitUntilInstanceRunning(r ->
r.instanceIds(instanceIdVal));
            System.out.println("Successfully started EC2 instance " +
instanceIdVal + " based on AMI " + amiId);
            return instanceIdVal;

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    // Get a list of instance types.
    public static String getInstanceTypes(Ec2Client ec2) {
        String instanceType;
        try {
            DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
                .maxResults(10)
                .build();

            DescribeInstanceTypesResponse response =
ec2.describeInstanceTypes(typesRequest);
            List<InstanceTypeInfo> instanceTypes = response.instanceTypes();
            for (InstanceTypeInfo type : instanceTypes) {
                System.out.println("The memory information of this type is " +
type.memoryInfo().sizeInMiB());
                System.out.println("Network information is " +
type.networkInfo().toString());
                System.out.println("Instance type is " +
type.instanceType().toString());
                instanceType = type.instanceType().toString();
                if (instanceType.compareTo("t2.2xlarge")  == 0){
                    return instanceType;
                }
            }
```

```
        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    // Display the Description field that corresponds to the instance Id value.
    public static String describeImage(Ec2Client ec2, String instanceId) {
        try {
            DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
                    .imageIds(instanceId)
                    .build();

            DescribeImagesResponse response = ec2.describeImages(imagesRequest);
            System.out.println("The description of the first image is " +
response.images().get(0).description());
            System.out.println("The name of the first image is " +
response.images().get(0).name());

            // Return the image Id value.
            return response.images().get(0).imageId();

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    // Get the Id value of an instance with amzn2 in the name.
    public static String getParaValues(SsmClient ssmClient) {
        try {
            GetParametersByPathRequest parameterRequest =
GetParametersByPathRequest.builder()
                    .path("/aws/service/ami-amazon-linux-latest")
                    .build();

            GetParametersByPathIterable responses =
ssmClient.getParametersByPathPaginator(parameterRequest);
            for
(software.amazon.awssdk.services.ssm.model.GetParametersByPathResponse
response : responses) {
                System.out.println("Test " + response.nextToken());
```

```
                List<Parameter> parameterList = response.parameters();
                for (Parameter para : parameterList) {
                    System.out.println("The name of the para is: " +
para.name());
                    System.out.println("The type of the para is: " +
para.type());
                    if (filterName(para.name())) {
                        return para.value();
                    }
                }
            }

        } catch (SsmException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
        return "";
    }

    // Return true if the name has amzn2 in it. For example:
    // /aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-arm64-gp2
    private static boolean filterName(String name) {
        String[] parts = name.split("/");
        String myValue = parts[4];
        return myValue.contains("amzn2");
    }

    public static void describeSecurityGroups(Ec2Client ec2, String groupId) {
        try {
            DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
                .groupIds(groupId)
                .build();

            // Use a paginator.
            DescribeSecurityGroupsIterable listGroups =
ec2.describeSecurityGroupsPaginator(request);
            listGroups.stream()
                .flatMap(r -> r.securityGroups().stream())
                .forEach(group -> System.out
                    .println(" Group id: " +group.groupId() + " group name = " +
group.groupName()));

        } catch (Ec2Exception e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String createSecurityGroup(Ec2Client ec2, String groupName,
String groupDesc, String vpcId,
            String myIpAddress) {
        try {
            CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
                    .groupName(groupName)
                    .description(groupDesc)
                    .vpcId(vpcId)
                    .build();

            CreateSecurityGroupResponse resp =
ec2.createSecurityGroup(createRequest);
            IpRange ipRange = IpRange.builder()
                    .cidrIp(myIpAddress + "/0")
                    .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();

            ec2.authorizeSecurityGroupIngress(authRequest);
```

```
            System.out.println("Successfully added ingress policy to security
group " + groupName);
            return resp.groupId();

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void describeKeys(Ec2Client ec2) {
        try {
            DescribeKeyPairsResponse response = ec2.describeKeyPairs();
            response.keyPairs().forEach(keyPair -> System.out.printf(
                    "Found key pair with name %s " +
                            "and fingerprint %s",
                    keyPair.keyName(),
                    keyPair.keyFingerprint()));

        } catch (Ec2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createKeyPair(Ec2Client ec2, String keyName, String
fileName) {
        try {
            CreateKeyPairRequest request = CreateKeyPairRequest.builder()
                    .keyName(keyName)
                    .build();

            CreateKeyPairResponse response = ec2.createKeyPair(request);
            String content = response.keyMaterial();
            BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
            writer.write(content);
            writer.close();
            System.out.println("Successfully created key pair named " + keyName);

        } catch (Ec2Exception | IOException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
```

```
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API 参考》中的以下主题。
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
import { mkdtempSync, writeFileSync, rmSync } from "fs";
import { tmpdir } from "os";
import { join } from "path";
import { get } from "http";

import {
  AllocateAddressCommand,
  AssociateAddressCommand,
  AuthorizeSecurityGroupIngressCommand,
  CreateKeyPairCommand,
  CreateSecurityGroupCommand,
  DeleteKeyPairCommand,
  DeleteSecurityGroupCommand,
  DescribeInstancesCommand,
  DescribeKeyPairsCommand,
  DescribeSecurityGroupsCommand,
  DisassociateAddressCommand,
  EC2Client,
  paginateDescribeImages,
  paginateDescribeInstanceTypes,
  ReleaseAddressCommand,
  RunInstancesCommand,
  StartInstancesCommand,
  StopInstancesCommand,
  TerminateInstancesCommand,
  waitUntilInstanceStatusOk,
  waitUntilInstanceStopped,
  waitUntilInstanceTerminated,
} from "@aws-sdk/client-ec2";
import { paginateGetParametersByPath, SSMClient } from "@aws-sdk/client-ssm";

import { wrapText } from "@aws-doc-sdk-examples/lib/utils/util-string.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";

const ec2Client = new EC2Client();
const ssmClient = new SSMClient();

const prompter = new Prompter();
const confirmMessage = "Continue?";
const tmpDirectory = mkdtempSync(join(tmpdir(), "ec2-scenario-tmp"));

const createKeyPair = async (keyPairName) => {
```

```
  // Create a key pair in Amazon EC2.
  const { KeyMaterial, KeyPairId } = await ec2Client.send(
    // A unique name for the key pair. Up to 255 ASCII characters.
    new CreateKeyPairCommand({ KeyName: keyPairName }),
  );

  // Save the private key in a temporary location.
  writeFileSync(`${tmpDirectory}/${keyPairName}.pem`, KeyMaterial, {
    mode: 0o400,
  });

  return KeyPairId;
};

const describeKeyPair = async (keyPairName) => {
  const command = new DescribeKeyPairsCommand({
    KeyNames: [keyPairName],
  });
  const { KeyPairs } = await ec2Client.send(command);
  return KeyPairs[0];
};

const createSecurityGroup = async (securityGroupName) => {
  const command = new CreateSecurityGroupCommand({
    GroupName: securityGroupName,
    Description: "A security group for the Amazon EC2 example.",
  });
  const { GroupId } = await ec2Client.send(command);
  return GroupId;
};

const allocateIpAddress = async () => {
  const command = new AllocateAddressCommand({});
  const { PublicIp, AllocationId } = await ec2Client.send(command);
  return { PublicIp, AllocationId };
};

const getLocalIpAddress = () => {
  return 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);
    });
  });
};

const authorizeSecurityGroupIngress = async (securityGroupId) => {
  const ipAddress = await getLocalIpAddress();
  const command = new AuthorizeSecurityGroupIngressCommand({
    GroupId: securityGroupId,
    IpPermissions: [
      {
        IpProtocol: "tcp",
        FromPort: 22,
        ToPort: 22,
        IpRanges: [{ CidrIp: `${ipAddress}/32` }],
      },
    ],
  });

  await ec2Client.send(command);
  return ipAddress;
};

const describeSecurityGroup = async (securityGroupName) => {
  const command = new DescribeSecurityGroupsCommand({
    GroupNames: [securityGroupName],
  });
  const { SecurityGroups } = await ec2Client.send(command);

  return SecurityGroups[0];
};

const getAmznLinux2AMIs = async () => {
  const AMIs = [];
  for await (const page of paginateGetParametersByPath(
    {
      client: ssmClient,
    },
    { Path: "/aws/service/ami-amazon-linux-latest" },
  )) {
    page.Parameters.forEach((param) => {
      if (param.Name.includes("amzn2")) {
        AMIs.push(param.Value);
      }
```

```
    });
  }

  const imageDetails = [];

  for await (const page of paginateDescribeImages(
    { client: ec2Client },
    { ImageIds: AMIs },
  )) {
    imageDetails.push(...(page.Images || []));
  }

  const choices = imageDetails.map((image, index) => ({
    name: `${image.ImageId} - ${image.Description}`,
    value: index,
  }));

  /**
   * @type {number}
   */
  const selectedIndex = await prompter.select({
    message: "Select an image.",
    choices,
  });

  return imageDetails[selectedIndex];
};

/**
 * @param {import('@aws-sdk/client-ec2').Image} imageDetails
 */
const getCompatibleInstanceTypes = async (imageDetails) => {
  const paginator = paginateDescribeInstanceTypes(
    { client: ec2Client, pageSize: 25 },
    {
      Filters: [
        {
          Name: "processor-info.supported-architecture",
          Values: [imageDetails.Architecture],
        },
        { Name: "instance-type", Values: ["*.micro", "*.small"] },
      ],
    },
  );
```

```
  const instanceTypes = [];

  for await (const page of paginator) {
    if (page.InstanceTypes.length) {
      instanceTypes.push(...(page.InstanceTypes || []));
    }
  }

  const choices = instanceTypes.map((type, index) => ({
    name: `${type.InstanceType} - Memory:${type.MemoryInfo.SizeInMiB}`,
    value: index,
  }));

  /**
   * @type {number}
   */
  const selectedIndex = await prompter.select({
    message: "Select an instance type.",
    choices,
  });
  return instanceTypes[selectedIndex];
};

const runInstance = async ({
  keyPairName,
  securityGroupId,
  imageId,
  instanceType,
}) => {
  const command = new RunInstancesCommand({
    KeyName: keyPairName,
    SecurityGroupIds: [securityGroupId],
    ImageId: imageId,
    InstanceType: instanceType,
    MinCount: 1,
    MaxCount: 1,
  });

  const { Instances } = await ec2Client.send(command);
  await waitUntilInstanceStatusOk(
    { client: ec2Client },
    { InstanceIds: [Instances[0].InstanceId] },
  );
```

```
  return Instances[0].InstanceId;
};

const describeInstance = async (instanceId) => {
  const command = new DescribeInstancesCommand({
    InstanceIds: [instanceId],
  });

  const { Reservations } = await ec2Client.send(command);
  return Reservations[0].Instances[0];
};

const displaySSHConnectionInfo = ({ publicIp, keyPairName }) => {
  return `ssh -i ${tmpDirectory}/${keyPairName}.pem ec2-user@${publicIp}`;
};

const stopInstance = async (instanceId) => {
  const command = new StopInstancesCommand({ InstanceIds: [instanceId] });
  await ec2Client.send(command);
  await waitUntilInstanceStopped(
    { client: ec2Client },
    { InstanceIds: [instanceId] },
  );
};

const startInstance = async (instanceId) => {
  const startCommand = new StartInstancesCommand({ InstanceIds: [instanceId] });
  await ec2Client.send(startCommand);
  await waitUntilInstanceStatusOk(
    { client: ec2Client },
    { InstanceIds: [instanceId] },
  );
  return await describeInstance(instanceId);
};

const associateAddress = async ({ allocationId, instanceId }) => {
  const command = new AssociateAddressCommand({
    AllocationId: allocationId,
    InstanceId: instanceId,
  });

  const { AssociationId } = await ec2Client.send(command);
  return AssociationId;
};
```

```
const disassociateAddress = async (associationId) => {
  const command = new DisassociateAddressCommand({
    AssociationId: associationId,
  });
  try {
    await ec2Client.send(command);
  } catch (err) {
    console.warn(
      `Failed to disassociated address with association id: ${associationId}`,
      err,
    );
  }
};

const releaseAddress = async (allocationId) => {
  const command = new ReleaseAddressCommand({
    AllocationId: allocationId,
  });

  try {
    await ec2Client.send(command);
    console.log(`Address with allocation ID ${allocationId} released.\n`);
  } catch (err) {
    console.log(
      `Failed to release address with allocation id: ${allocationId}.`,
      err,
    );
  }
};

const restartInstance = async (instanceId) => {
  console.log("Stopping instance.");
  await stopInstance(instanceId);
  console.log("Instance stopped.");
  console.log("Starting instance.");
  const { PublicIpAddress } = await startInstance(instanceId);
  return PublicIpAddress;
};

const terminateInstance = async (instanceId) => {
  const command = new TerminateInstancesCommand({
    InstanceIds: [instanceId],
  });
```

```
  try {
    await ec2Client.send(command);
    await waitUntilInstanceTerminated(
      { client: ec2Client },
      { InstanceIds: [instanceId] },
    );
    console.log(`Instance with ID ${instanceId} terminated.\n`);
  } catch (err) {
    console.warn(`Failed to terminate instance ${instanceId}.`, err);
  }
};

const deleteSecurityGroup = async (securityGroupId) => {
  const command = new DeleteSecurityGroupCommand({
    GroupId: securityGroupId,
  });

  try {
    await ec2Client.send(command);
    console.log(`Security group ${securityGroupId} deleted.\n`);
  } catch (err) {
    console.warn(`Failed to delete security group ${securityGroupId}.`, err);
  }
};

const deleteKeyPair = async (keyPairName) => {
  const command = new DeleteKeyPairCommand({
    KeyName: keyPairName,
  });

  try {
    await ec2Client.send(command);
    console.log(`Key pair ${keyPairName} deleted.\n`);
  } catch (err) {
    console.warn(`Failed to delete key pair ${keyPairName}.`, err);
  }
};

const deleteTemporaryDirectory = () => {
  try {
    rmSync(tmpDirectory, { recursive: true });
    console.log(`Temporary directory ${tmpDirectory} deleted.\n`);
  } catch (err) {
```

```
    console.warn(`Failed to delete temporary directory ${tmpDirectory}.`, err);
  }
};

export const main = async () => {
  const keyPairName = "ec2-scenario-key-pair";
  const securityGroupName = "ec2-scenario-security-group";

  let securityGroupId, ipAllocationId, publicIp, instanceId, associationId;

  console.log(wrapText("Welcome to the Amazon EC2 basic usage scenario."));

  try {
    // Prerequisites
    console.log(
      "Before you launch an instance, you'll need a few things:",
      "\n - A Key Pair",
      "\n - A Security Group",
      "\n - An IP Address",
      "\n - An AMI",
      "\n - A compatible instance type",
      "\n\n I'll go ahead and take care of the first three, but I'll need your
 help for the rest.",
    );

    await prompter.confirm({ message: confirmMessage });

    await createKeyPair(keyPairName);
    securityGroupId = await createSecurityGroup(securityGroupName);
    const { PublicIp, AllocationId } = await allocateIpAddress();
    ipAllocationId = AllocationId;
    publicIp = PublicIp;
    const ipAddress = await authorizeSecurityGroupIngress(securityGroupId);

    const { KeyName } = await describeKeyPair(keyPairName);
    const { GroupName } = await describeSecurityGroup(securityGroupName);
    console.log(`# created the key pair ${KeyName}.\n`);
    console.log(
      `# created the security group ${GroupName}`,
      `and allowed SSH access from ${ipAddress} (your IP).\n`,
    );
    console.log(`# allocated ${publicIp} to be used for your EC2 instance.\n`);

    await prompter.confirm({ message: confirmMessage });
```

```
  // Creating the instance
  console.log(wrapText("Create the instance."));
  console.log(
    "You get to choose which image you want. Select an amazon-linux-2 image
from the following:",
  );
  const imageDetails = await getAmznLinux2AMIs();
  const instanceTypeDetails = await getCompatibleInstanceTypes(imageDetails);
  console.log("Creating your instance. This can take a few seconds.");
  instanceId = await runInstance({
    keyPairName,
    securityGroupId,
    imageId: imageDetails.ImageId,
    instanceType: instanceTypeDetails.InstanceType,
  });
  const instanceDetails = await describeInstance(instanceId);
  console.log(`# instance ${instanceId}.\n`);
  console.log(instanceDetails);
  console.log(
    `\nYou should now be able to SSH into your instance from another
terminal:`,
    `\n${displaySSHConnectionInfo({
      publicIp: instanceDetails.PublicIpAddress,
      keyPairName,
    })}`,
  );

  await prompter.confirm({ message: confirmMessage });

  // Understanding the IP address.
  console.log(wrapText("Understanding the IP address."));
  console.log(
    "When you stop and start an instance, the IP address will change. I'll
restart your",
    "instance for you. Notice how the IP address changes.",
  );
  const ipAddressAfterRestart = await restartInstance(instanceId);
  console.log(
    `\n Instance started. The IP address changed from
${instanceDetails.PublicIpAddress} to ${ipAddressAfterRestart}`,
    `\n${displaySSHConnectionInfo({
      publicIp: ipAddressAfterRestart,
      keyPairName,
```

```
      })}`,
    );
    await prompter.confirm({ message: confirmMessage });
    console.log(
      `If you want to the IP address to be static, you can associate an
allocated`,
      `IP address to your instance. I allocated ${publicIp} for you earlier, and
now I'll associate it to your instance.`,
    );
    associationId = await associateAddress({
      allocationId: ipAllocationId,
      instanceId,
    });
    console.log(
      "Done. Now you should be able to SSH using the new IP.\n",
      `${displaySSHConnectionInfo({ publicIp, keyPairName })}`,
    );
    await prompter.confirm({ message: confirmMessage });
    console.log(
      "I'll restart the server again so you can see the IP address remains the
same.",
    );
    const ipAddressAfterAssociated = await restartInstance(instanceId);
    console.log(
      `Done. Here's your SSH info. Notice the IP address hasn't changed.`,
      `\n${displaySSHConnectionInfo({
        publicIp: ipAddressAfterAssociated,
        keyPairName,
      })}`,
    );
    await prompter.confirm({ message: confirmMessage });
  } catch (err) {
    console.error(err);
  } finally {
    // Clean up.
    console.log(wrapText("Clean up."));
    console.log("Now I'll clean up all of the stuff I created.");
    await prompter.confirm({ message: confirmMessage });
    console.log("Cleaning up. Some of these steps can take a bit of time.");
    await disassociateAddress(associationId);
    await terminateInstance(instanceId);
    await releaseAddress(ipAllocationId);
    await deleteSecurityGroup(securityGroupId);
    deleteTemporaryDirectory();
```

```
    await deleteKeyPair(keyPairName);
    console.log(
      "Done cleaning up. Thanks for staying until the end!",
      "If you have any feedback please use the feedback button in the docs",
      "or create an issue on GitHub.",
    );
  }
};
```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API 参考》中的以下主题。
  - AllocateAddress
  - AssociateAddress
  - AuthorizeSecurityGroupIngress
  - CreateKeyPair
  - CreateSecurityGroup
  - DeleteKeyPair
  - DeleteSecurityGroup
  - DescribeImages
  - DescribeInstanceTypes
  - DescribeInstances
  - DescribeKeyPairs
  - DescribeSecurityGroups
  - DisassociateAddress
  - ReleaseAddress
  - RunInstances
  - StartInstances
  - StopInstances
  - TerminateInstances
  - UnmonitorInstances

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

```
/**
 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

 This Kotlin 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 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.waitUntilInstanceTerminated {
            // 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.waitUntilInstanceRunning {
            // 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.waitUntilInstanceStopped {
            // 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")
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Kotlin API 参考》中的以下主题。
  - AllocateAddress
  - AssociateAddress

- AuthorizeSecurityGroupIngress
- CreateKeyPair
- CreateSecurityGroup
- DeleteKeyPair
- DeleteSecurityGroup
- DescribeImages
- DescribeInstanceTypes
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DisassociateAddress
- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

Python

SDK for Python (Boto3)

> **Note**
>
> 还有更多相关信息 GitHub。在 AWS 代码示例存储库中查找完整实例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
class Ec2InstanceScenario:
    """Runs an interactive scenario that shows how to get started using EC2
 instances."""
```

```
    def __init__(self, inst_wrapper, key_wrapper, sg_wrapper, eip_wrapper,
ssm_client):
        """
        :param inst_wrapper: An object that wraps instance actions.
        :param key_wrapper: An object that wraps key pair actions.
        :param sg_wrapper: An object that wraps security group actions.
        :param eip_wrapper: An object that wraps Elastic IP actions.
        :param ssm_client: A Boto3 AWS Systems Manager client.
        """
        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

    @demo_func
    def create_and_list_key_pairs(self):
        """
        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. Lists the first five key pairs for the current account.
        """
        print(
            "Let's create an RSA key pair that you can be use to securely connect
to "
            "your EC2 instance."
        )
        key_name = q.ask("Enter a unique name for your key: ", q.non_empty)
        self.key_wrapper.create(key_name)
        print(
            f"Created a key pair {self.key_wrapper.key_pair.key_name} and saved
the "
            f"private key to {self.key_wrapper.key_file_path}.\n"
        )
        if q.ask("Do you want to list some of your key pairs? (y/n) ",
q.is_yesno):
            self.key_wrapper.list(5)

    @demo_func
    def create_security_group(self):
        """
        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.
        """
        print("Let's create a security group to manage access to your instance.")
        sg_name = q.ask("Enter a unique name for your security group: ",
q.non_empty)
        security_group = self.sg_wrapper.create(
            sg_name, "Security group for example: get started with instances."
        )
        print(
            f"Created security group {security_group.group_name} in your default
"
            f"VPC {security_group.vpc_id}.\n"
        )

        ip_response = urllib.request.urlopen("http://checkip.amazonaws.com")
        current_ip_address = ip_response.read().decode("utf-8").strip()
        print("Let's add a rule to allow SSH only from your current IP address.")
        print(f"Your public IP address is {current_ip_address}.")
        q.ask("Press Enter to add this rule to your security group.")
        response = self.sg_wrapper.authorize_ingress(current_ip_address)
        if response["Return"]:
            print("Security group rules updated.")
        else:
            print("Couldn't update security group rules.")
        self.sg_wrapper.describe()

    @demo_func
    def create_instance(self):
        """
        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.
        """
        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"]]
        )
        print(
            "Let's create an instance from an Amazon Linux 2 AMI. Here are some
 options:"
        )
        image_choice = q.choose(
            "Which one do you want to use? ", [opt.description for opt in
 amzn2_images]
        )
        print("Great choice!\n")

        print(
            f"Here are some instance types that support the "
            f"{amzn2_images[image_choice].architecture} architecture of the
 image:"
        )
        inst_types = self.inst_wrapper.get_instance_types(
            amzn2_images[image_choice].architecture
        )
        inst_type_choice = q.choose(
            "Which one do you want to use? ", [it["InstanceType"] for it in
 inst_types]
        )
        print("Another great choice.\n")
```

```
        print("Creating your instance and waiting for it to start...")
        self.inst_wrapper.create(
            amzn2_images[image_choice],
            inst_types[inst_type_choice]["InstanceType"],
            self.key_wrapper.key_pair,
            [self.sg_wrapper.security_group],
        )
        print(f"Your instance is ready:\n")
        self.inst_wrapper.display()

        print("You can use SSH to connect to your instance.")
        print(
            "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):
        """
        Displays an SSH connection string that can be used to connect to a
running
        instance.
        """
        print("To connect, open another command prompt and run the following
command:")
        if self.eip_wrapper.elastic_ip is None:
            print(
                f"\tssh -i {self.key_wrapper.key_file_path} "
                f"ec2-user@{self.inst_wrapper.instance.public_ip_address}"
            )
        else:
            print(
                f"\tssh -i {self.key_wrapper.key_file_path} "
                f"ec2-user@{self.eip_wrapper.elastic_ip.public_ip}"
            )
        q.ask("Press Enter when you're ready to continue the demo.")

    @demo_func
    def associate_elastic_ip(self):
        """
        1. Allocates an Elastic IP address and associates it with the instance.
```

```
        2. Displays an SSH connection string that uses the Elastic IP address.
        """
        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."
        )
        elastic_ip = self.eip_wrapper.allocate()
        print(f"Allocated static Elastic IP address: {elastic_ip.public_ip}.")
        self.eip_wrapper.associate(self.inst_wrapper.instance)
        print(f"Associated your Elastic IP with your instance.")
        print(
            "You can now use SSH to connect to your instance by using the Elastic
IP."
        )
        self._display_ssh_info()

    @demo_func
    def stop_and_start_instance(self):
        """
        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.
        """
        print("Let's stop and start your instance to see what changes.")
        print("Stopping your instance and waiting until it's stopped...")
        self.inst_wrapper.stop()
        print("Your instance is stopped. Restarting...")
        self.inst_wrapper.start()
        print("Your instance is running.")
        self.inst_wrapper.display()
        if self.eip_wrapper.elastic_ip is None:
            print(
                "Every time your instance is restarted, its public IP address
changes."
            )
        else:
            print(
```

```
                "Because you have associated an Elastic IP with your instance,
you can \n"
                "connect by using a consistent IP address after the instance
restarts."
            )
        self._display_ssh_info()

    @demo_func
    def cleanup(self):
        """
        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.
        """
        print("Let's clean everything up. This example created these resources:")
        print(f"\tElastic IP: {self.eip_wrapper.elastic_ip.allocation_id}")
        print(f"\tInstance: {self.inst_wrapper.instance.id}")
        print(f"\tSecurity group: {self.sg_wrapper.security_group.id}")
        print(f"\tKey pair: {self.key_wrapper.key_pair.name}")
        if q.ask("Ready to delete these resources? (y/n) ", q.is_yesno):
            self.eip_wrapper.disassociate()
            print("Disassociated the Elastic IP from the instance.")
            self.eip_wrapper.release()
            print("Released the Elastic IP.")
            print("Terminating the instance and waiting for it to terminate...")
            self.inst_wrapper.terminate()
            print("Instance terminated.")
            self.sg_wrapper.delete()
            print("Deleted security group.")
            self.key_wrapper.delete()
            print("Deleted key pair.")

    def run_scenario(self):
        logging.basicConfig(level=logging.INFO, format="%(levelname)s:
%(message)s")

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

        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = Ec2InstanceScenario(
            InstanceWrapper.from_resource(),
            KeyPairWrapper.from_resource(),
            SecurityGroupWrapper.from_resource(),
            ElasticIpWrapper.from_resource(),
            boto3.client("ssm"),
        )
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

定义一个包装密钥对操作的类。

```
class KeyPairWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair
 actions."""

    def __init__(self, ec2_resource, key_file_dir, key_pair=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
 resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param key_file_dir: The folder where the private key information is
 stored.
                             This should be a secure folder.
        :param key_pair: A Boto3 KeyPair object. This is a high-level object that
                         wraps key pair actions.
```

```
        """
        self.ec2_resource = ec2_resource
        self.key_pair = key_pair
        self.key_file_path = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource, tempfile.TemporaryDirectory())


    def create(self, key_name):
        """
        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 Boto3 KeyPair object that represents the newly created key
pair.
        """
        try:
            self.key_pair = self.ec2_resource.create_key_pair(KeyName=key_name)
            self.key_file_path = os.path.join(
                self.key_file_dir.name, f"{self.key_pair.name}.pem"
            )
            with open(self.key_file_path, "w") as key_file:
                key_file.write(self.key_pair.key_material)
        except ClientError as err:
            logger.error(
                "Couldn't create key %s. Here's why: %s: %s",
                key_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.key_pair


    def list(self, limit):
```

```
        """
        Displays a list of key pairs for the current account.

        :param limit: The maximum number of key pairs to list.
        """
        try:
            for kp in self.ec2_resource.key_pairs.limit(limit):
                print(f"Found {kp.key_type} key {kp.name} with fingerprint:")
                print(f"\t{kp.key_fingerprint}")
        except ClientError as err:
            logger.error(
                "Couldn't list key pairs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete(self):
        """
        Deletes a key pair.
        """
        if self.key_pair is None:
            logger.info("No key pair to delete.")
            return

        key_name = self.key_pair.name
        try:
            self.key_pair.delete()
            self.key_pair = None
        except ClientError as err:
            logger.error(
                "Couldn't delete key %s. Here's why: %s : %s",
                key_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

定义一个包装安全组操作的类。

```
class SecurityGroupWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) security group
 actions."""

    def __init__(self, ec2_resource, security_group=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
 resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param security_group: A Boto3 SecurityGroup object. This is a high-level
 object
                               that wraps security group actions.
        """
        self.ec2_resource = ec2_resource
        self.security_group = security_group

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource)


    def create(self, group_name, group_description):
        """
        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: A Boto3 SecurityGroup object that represents the newly created
 security group.
        """
        try:
            self.security_group = self.ec2_resource.create_security_group(
                GroupName=group_name, Description=group_description
            )
        except ClientError as err:
            logger.error(
                "Couldn't create security group %s. Here's why: %s: %s",
```

```
                group_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.security_group


    def authorize_ingress(self, ssh_ingress_ip):
        """
        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.
        """
        if self.security_group is None:
            logger.info("No security group to update.")
            return

        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.security_group.authorize_ingress(
                IpPermissions=ip_permissions
            )
        except ClientError as err:
            logger.error(
                "Couldn't authorize inbound rules for %s. Here's why: %s: %s",
                self.security_group.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
```

```
            raise
        else:
            return response


    def describe(self):
        """
        Displays information about the security group.
        """
        if self.security_group is None:
            logger.info("No security group to describe.")
            return

        try:
            print(f"Security group: {self.security_group.group_name}")
            print(f"\tID: {self.security_group.id}")
            print(f"\tVPC: {self.security_group.vpc_id}")
            if self.security_group.ip_permissions:
                print(f"Inbound permissions:")
                pp(self.security_group.ip_permissions)
        except ClientError as err:
            logger.error(
                "Couldn't get data for security group %s. Here's why: %s: %s",
                self.security_group.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete(self):
        """
        Deletes the security group.
        """
        if self.security_group is None:
            logger.info("No security group to delete.")
            return

        group_id = self.security_group.id
        try:
            self.security_group.delete()
        except ClientError as err:
            logger.error(
                "Couldn't delete security group %s. Here's why: %s: %s",
```

```
                group_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

定义一个包装实例操作的类。

```
class InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance
 actions."""

    def __init__(self, ec2_resource, instance=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
 resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param instance: A Boto3 Instance object. This is a high-level object
 that
                          wraps instance actions.
        """
        self.ec2_resource = ec2_resource
        self.instance = instance

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource)


    def create(self, image, instance_type, key_pair, security_groups=None):
        """
        Creates a new EC2 instance. The instance starts immediately after
        it is created.

        The instance is created in the default VPC of the current account.
```

```
        :param image: A Boto3 Image object that represents an Amazon Machine
Image (AMI)
                      that defines attributes of the instance that is created.
The AMI
                      defines things like the kind of operating system and the
type of
                      storage used by the instance.
        :param instance_type: The type of instance to create, such as 't2.micro'.
                              The instance type defines things like the number of
CPUs and
                              the amount of memory.
        :param key_pair: A Boto3 KeyPair or KeyPairInfo object that represents
the key
                         pair that is used to secure connections to the instance.
        :param security_groups: A list of Boto3 SecurityGroup objects that
represents the
                                security groups that are used to grant access to
the
                                instance. When no security groups are specified,
the
                                default security group of the VPC is used.
        :return: A Boto3 Instance object that represents the newly created
instance.
        """
        try:
            instance_params = {
                "ImageId": image.id,
                "InstanceType": instance_type,
                "KeyName": key_pair.name,
            }
            if security_groups is not None:
                instance_params["SecurityGroupIds"] = [sg.id for sg in
security_groups]
            self.instance = self.ec2_resource.create_instances(
                **instance_params, MinCount=1, MaxCount=1
            )[0]
            self.instance.wait_until_running()
        except ClientError as err:
            logging.error(
                "Couldn't create instance with image %s, instance type %s, and
key %s. "
                "Here's why: %s: %s",
                image.id,
                instance_type,
```

```
                key_pair.name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.instance


    def display(self, indent=1):
        """
        Displays information about an instance.

        :param indent: The visual indent to apply to the output.
        """
        if self.instance is None:
            logger.info("No instance to display.")
            return

        try:
            self.instance.load()
            ind = "\t" * indent
            print(f"{ind}ID: {self.instance.id}")
            print(f"{ind}Image ID: {self.instance.image_id}")
            print(f"{ind}Instance type: {self.instance.instance_type}")
            print(f"{ind}Key name: {self.instance.key_name}")
            print(f"{ind}VPC ID: {self.instance.vpc_id}")
            print(f"{ind}Public IP: {self.instance.public_ip_address}")
            print(f"{ind}State: {self.instance.state['Name']}")
        except ClientError as err:
            logger.error(
                "Couldn't display your instance. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def terminate(self):
        """
        Terminates an instance and waits for it to be in a terminated state.
        """
        if self.instance is None:
            logger.info("No instance to terminate.")
```

```
            return

        instance_id = self.instance.id
        try:
            self.instance.terminate()
            self.instance.wait_until_terminated()
            self.instance = None
        except ClientError as err:
            logging.error(
                "Couldn't terminate instance %s. Here's why: %s: %s",
                instance_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def start(self):
        """
        Starts an instance and waits for it to be in a running state.

        :return: The response to the start request.
        """
        if self.instance is None:
            logger.info("No instance to start.")
            return

        try:
            response = self.instance.start()
            self.instance.wait_until_running()
        except ClientError as err:
            logger.error(
                "Couldn't start instance %s. Here's why: %s: %s",
                self.instance.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def stop(self):
        """
```

```
        Stops an instance and waits for it to be in a stopped state.

        :return: The response to the stop request.
        """
        if self.instance is None:
            logger.info("No instance to stop.")
            return

        try:
            response = self.instance.stop()
            self.instance.wait_until_stopped()
        except ClientError as err:
            logger.error(
                "Couldn't stop instance %s. Here's why: %s: %s",
                self.instance.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response


    def get_images(self, image_ids):
        """
        Gets information about Amazon Machine Images (AMIs) from a list of AMI
IDs.

        :param image_ids: The list of AMIs to look up.
        :return: A list of Boto3 Image objects that represent the requested AMIs.
        """
        try:
            images = list(self.ec2_resource.images.filter(ImageIds=image_ids))
        except ClientError as err:
            logger.error(
                "Couldn't get images. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return images
```

```
   def get_instance_types(self, architecture):
       """
       Gets instance types that support the specified architecture and are
designated
       as either 'micro' or 'small'. When an instance is created, the instance
type
       you specify must support the architecture of the AMI you use.

       :param architecture: The kind of architecture the instance types must
support,
                            such as 'x86_64'.
       :return: A list of instance types that support the specified architecture
                and are either 'micro' or 'small'.
       """
       try:
           inst_types = []
           it_paginator = self.ec2_resource.meta.client.get_paginator(
               "describe_instance_types"
           )
           for page in it_paginator.paginate(
               Filters=[
                   {
                       "Name": "processor-info.supported-architecture",
                       "Values": [architecture],
                   },
                   {"Name": "instance-type", "Values": ["*.micro", "*.small"]},
               ]
           ):
               inst_types += page["InstanceTypes"]
       except ClientError as err:
           logger.error(
               "Couldn't get instance types. Here's why: %s: %s",
               err.response["Error"]["Code"],
               err.response["Error"]["Message"],
           )
           raise
       else:
           return inst_types
```

定义一个包装弹性 IP 操作的类。

```
class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
 actions."""

    def __init__(self, ec2_resource, elastic_ip=None):
        """
        :param ec2_resource: A Boto3 Amazon EC2 resource. This high-level
 resource
                             is used to create additional high-level objects
                             that wrap low-level Amazon EC2 service actions.
        :param elastic_ip: A Boto3 VpcAddress object. This is a high-level object
 that
                           wraps Elastic IP actions.
        """
        self.ec2_resource = ec2_resource
        self.elastic_ip = elastic_ip

    @classmethod
    def from_resource(cls):
        ec2_resource = boto3.resource("ec2")
        return cls(ec2_resource)


    def allocate(self):
        """
        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 newly created Elastic IP object. By default, the address is
 not
                 associated with any instance.
        """
        try:
            response =
 self.ec2_resource.meta.client.allocate_address(Domain="vpc")
            self.elastic_ip =
 self.ec2_resource.VpcAddress(response["AllocationId"])
        except ClientError as err:
            logger.error(
                "Couldn't allocate Elastic IP. Here's why: %s: %s",
```

```
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.elastic_ip


    def associate(self, instance):
        """
        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 instance: A Boto3 Instance object. This is a high-level object
that wraps
                         Amazon EC2 instance actions.
        :return: A response that contains the ID of the association.
        """
        if self.elastic_ip is None:
            logger.info("No Elastic IP to associate.")
            return

        try:
            response = self.elastic_ip.associate(InstanceId=instance.id)
        except ClientError as err:
            logger.error(
                "Couldn't associate Elastic IP %s with instance %s. Here's why:
%s: %s",
                self.elastic_ip.allocation_id,
                instance.id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        return response


    def disassociate(self):
        """
        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.
        """
        if self.elastic_ip is None:
            logger.info("No Elastic IP to disassociate.")
            return

        try:
            self.elastic_ip.association.delete()
        except ClientError as err:
            logger.error(
                "Couldn't disassociate Elastic IP %s from its instance. Here's
why: %s: %s",
                self.elastic_ip.allocation_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def release(self):
        """
        Releases an Elastic IP address. After the Elastic IP address is released,
        it can no longer be used.
        """
        if self.elastic_ip is None:
            logger.info("No Elastic IP to release.")
            return

        try:
            self.elastic_ip.release()
        except ClientError as err:
            logger.error(
                "Couldn't release Elastic IP address %s. Here's why: %s: %s",
                self.elastic_ip.allocation_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API 参考》中的以下主题。

- AllocateAddress
- AssociateAddress
- AuthorizeSecurityGroupIngress
- CreateKeyPair
- CreateSecurityGroup
- DeleteKeyPair
- DeleteSecurityGroup
- DescribeImages
- DescribeInstanceTypes
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DisassociateAddress
- ReleaseAddress
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- UnmonitorInstances

有关 S AWS DK 开发者指南和代码示例的完整列表，请参阅使用 AWS 软件开发工具包创建 Amazon EC2 资源。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用亚马逊监控亚马逊 EC2 API 请求 CloudWatch

您可以使用 Amazon 监控 Amazon EC2 API 请求 CloudWatch，亚马逊会收集原始数据并将其处理成可读的近乎实时的指标。这些指标提供了一种简单的方法来跟踪 Amazon EC2 API 操作在一段时间内的使用情况和结果。这些信息使您可以更好地了解 Web 应用程序的性能，并使您能够识别和诊断各种问题。您还可以设置警报，监视某些阈值，并在达到这些阈值时发送通知或采取特定操作。

有关的更多信息 CloudWatch，请参阅 Amazon CloudWatch 用户指南。

> **Important**
>
> 亚马逊 EC2 API 指标是一项可选功能。您必须申请访问此功能。有关更多信息，请参阅 the section called “启用亚马逊 EC2 API 指标”。

内容

启用亚马逊 EC2 API 指标

使用以下步骤为您申请访问此功能的权限 AWS 账户。

请求访问此功能

1. 打开AWS Support 中心。
2. 选择创建案例。
3. 选择账户和账单。
4. 对于 “服务”，选择 “一般信息” 和 “入门”。
5. 在 “类别” 中，选择 “使用 AWS 和服务”。
6. 选择 Next step: Additional information（下一步：其他信息）。

7. 对于 Subject (主题)，请输入 **Request access to Amazon EC2 API metrics**。
8. 对于描述，输入 **Please grant my account access to Amazon EC2 API metrics. Related page: https://docs.aws.amazon.com/AWSEC2/latest/APIReference/monitor.html**。还要包括您需要访问的区域。
9. 选择下一步：立即解决或联系我们。
10. 在 “联系我们” 选项卡上，选择您的首选联系语言和联系方式。
11. 选择提交。

亚马逊 EC2 API 指标和维度

指标

Amazon EC2 API 指标包含在`AWS/EC2/API`命名空间中。下表列出了可用于 Amazon EC2 API 请求的指标。

指标	描述
`ClientErrors`	由客户端错误导致的 API 请求失败的数量。 这些错误通常是由客户端的行为引起的，例如在请求中指定了不正确或无效的参数，或者代表无权使用操作或资源的用户使用操作或资源。 单位：计数
`RequestLimitExceeded`	您的账户超过 Amazon EC2 API 允许的最大请求速率的次数。 Amazon EC2 API 请求会受到限制，以帮助维持服务的性能。如果您的请求已被限制，则会出现错误。`Client.RequestLimitExceeded` 单位：计数
`ServerErrors`	由内部服务器错误导致的 API 请求失败的数量。 这些错误通常是由 AWS 服务器端错误、异常或故障引起的。

指标	描述
	单位：计数
SuccessfulCalls	成功的 API 请求数。 单位：计数

尺寸

可以在所有 EC2 API 操作中筛选亚马逊 EC2 指标数据。有关尺寸的更多信息，请参阅 Amazon CloudWatch 概念。

指标数据保留

Amazon EC2 API 指标每隔 1 分钟发送一次。 CloudWatch CloudWatch 按如下方式保留指标数据：

- 时间段为 60 秒（1 分钟）的数据点可用 15 天。
- 周期为 300 秒（5 分钟）的数据点可用 63 天。
- 周期为 3600 秒（1 小时）的数据点可用 455 天（15 个月）。

监控以您的名义提出的请求

AWS 服务代表您发出的 API 请求（例如服务相关角色发出的请求）不计入您的 API 限制限制，也不会向亚马逊发送您账户 CloudWatch 的指标。无法使用监控这些请求 CloudWatch。

第三方服务提供商代表您发出的 API 请求会计入您的 API 限制限制，并且它们会向亚马逊发送您账户 CloudWatch 的指标。可以使用监控这些请求 CloudWatch。

Billing

适用标准 CloudWatch 定价和费用。使用 Amazon EC2 API 指标不收取任何额外费用。有关更多信息，请参阅 Amazon CloudWatch 定价。

与亚马逊合作 CloudWatch

目录

- 查看 CloudWatch 指标
- 创建 CloudWatch 警报

查看 CloudWatch 指标

使用以下过程查看 Amazon EC2 API 指标。

先决条件

您必须允许访问您的账户的 Amazon EC2 API 指标。有关更多信息，请参阅 the section called “启用亚马逊 EC2 API 指标”。

使用控制台查看 Amazon EC2 API 指标

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择指标，所有指标。
3. 在 “浏览” 选项卡上，选择 EC2/API 指标命名空间。
4. 要查看指标，请选择指标维度。

使用命令行查看 Amazon EC2 API 指标

使用以下命令之一：

- 列表指标 ()AWS CLI

```
aws cloudwatch list-metrics --namespace "AWS/EC2/API"
```

- get-cw () MetricListAWS Tools for Windows PowerShell

```
Get-CWMetricList -Namespace "AWS/EC2/API"
```

创建 CloudWatch 警报

您可以创建一个 CloudWatch 警报，当警报状态发生变化时，该警报会发送 Amazon SNS 消息。告警会监控您指定的时间段内的某个指标。它根据指标在多个时间段内相对于给定阈值的值向 SNS 主题发送通知。

例如，您可以创建一个警报，监控由于服务器端错误而失败 DescribeInstances的 API 请求数量。当 DescribeInstances API 请求失败次数在 5 分钟内达到 10 个服务器端错误的阈值时，以下警报会发送电子邮件通知。

先决条件

您必须允许访问您的账户的 Amazon EC2 API 指标。有关更多信息，请参阅 the section called “启用亚马逊 EC2 API 指标”。

要为 Amazon EC2 DescribeInstances API 创建警报，请请求服务器错误

1. 打开 CloudWatch 控制台，网址为 https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，依次选择 Alarms（警报）和 All alarms（所有警报）。
3. 选择Create alarm（创建警报）。
4. 选择选择指标，然后指定以下内容：
   a. 选择 EC2/API。
   b. 选择 “每项操作指标”。
   c. 选中与ServerErrors指标名称位于同一行旁边的复选框。DescribeInstances
   d. 选择选择指标。
5. 将显示 Specify metric and conditions（指定指标和条件）页面，其中显示一个图表以及有关您选择的指标和统计数据的其他信息。
   a. 在 “指标” 下，指定以下内容：
      i. 对于 Statistic（统计数据），选择 Sum（总计）。
      ii. 在 “时段” 中，确认已选择 5 分钟。
   b. 在条件下面，指定以下内容：
      i. 对于 Threshold type（阈值类型），选择 Static（静态）。
      ii. 对于 “无论何时 ServerErrors 是”，选择 “大于/等于 >=”。
      iii. 不止于... ，输入 10。
   c. 选择下一步。
6. Configure actions（配置操作）页面会显示。
   - 在 “通知” 下，指定以下内容：

i. 对于 Alam 状态触发器，请选择处于警报状态。

ii. 对于选择 SNS 主题，选择选择现有 SNS 主题或创建新主题，然后填写通知的必填字段。

iii. 选择下一步。

7. 将出现 “添加姓名和描述” 页面。

a. 在警报名称中，输入警报的名称。名称只能包含 ASCII 字符。

b. 在警报描述中，输入警报的可选描述。

c. 选择下一步。

8. 将出现 “预览并创建” 页面。验证信息是否正确，然后选择创建警报。

有关更多信息，请参阅亚马逊 CloudWatch 用户指南中的使用亚马逊 CloudWatch 警报。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。