



开发人员指南

Amazon Elastic Compute Cloud



Amazon Elastic Compute Cloud: 开发人员指南

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

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆或者贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Table of Contents

以编程方式访问 Amazon EC2	1
服务端点	1
IPv4 端点	2
双栈 (IPv4 和 IPv6) 端点	2
按区域划分的服务端点	2
指定端点	7
最终一致性	9
幂等性	10
Amazon EC2 中的幂等性	11
RunInstances 的幂等性	14
示例	15
针对幂等请求的重试建议	17
API 请求节流	17
如何应用节流	18
请求令牌限制	19
资源令牌限制	29
监控 API 节流	30
重试和指数回退	30
请求提高限制	30
分页	32
最佳实践	32
常见问题	33
使用 AWS CLI	36
了解有关 AWS CLI 的更多信息	36
使用 CloudFormation	37
Amazon EC2 和 CloudFormation 模板	37
Amazon EC2 的资源	37
了解有关 CloudFormation 的更多信息	40
使用 AWS 软件开发工具包	42
Amazon EC2 API 的代码示例	42
详细了解 AWS SDK	42
Amazon EC2 的低级别 API	43
Console-to-Code	44
代码示例	45

基本功能	52
你好 Amazon EC2	58
了解基本功能	75
操作	320
场景	1198
构建和管理弹性服务	1198
创建具有私有子网和 NAT 网关的 VPC	1366
开始使用 Amazon VPC	1383
中转网关入门	1399
VPC IPAM 入门	1407
使用监控 API 请求 CloudWatch	1418
启用亚马逊 EC2 API 指标	1418
亚马逊 EC2 API 指标和维度	1419
指标	1419
Dimensions	1420
指标数据留存	1420
监控代表您发起的请求	1420
计费	1421
与亚马逊合作 CloudWatch	1421
查看 CloudWatch 指标	1421
创建 CloudWatch 警报	1422
.....	mcdxxiv

以编程方式访问 Amazon EC2

您可以使用 AWS Management Console 或编程接口来创建和管理 Amazon EC2 资源。关于使用 Amazon EC2 控制台的相关信息，请参阅 [《Amazon EC2 用户指南》](#)。

工作方式

- [Amazon EC2 端点](#)
- [最终一致性](#)。
- [幂等性](#)
- [请求节流](#)
- [分页](#)

编程接口

- [AWS Command Line Interface \(AWS CLI\)](#)
- [AWS CloudFormation](#)
- [AWS SDK](#)
- [低级别 API](#)

入门

- [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](#)
- [双栈端点](#) (同时支持 IPv4 和 IPv6)
- [FIPS 端点](#)

当您发出请求时，您可以指定要使用的端点。如果您未指定终端节点，则默认使用该 IPv4 终端节点。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](#)。有关可用端点的列表，请参阅[按区域划分的服务端点](#)。

IPv4 端点

IPv4 端点仅支持 IPv4 流量。IPv4 终端节点适用于所有区域。

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，该 `eu-west-1` 区域的 IPv4 终端节点名称为 `ec2.eu-west-1.amazonaws.com`。

双栈 (IPv4 和 IPv6) 端点

双栈端点同时支持 IPv4 和 IPv6 流量。当您向双栈终端节点发出请求时，终端节点 URL 会解析为 IPv6 或 IPv4 地址，具体取决于您的网络和客户端使用的协议。

Amazon 仅 EC2 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`。

按区域划分的服务端点

以下是 Amazon 的服务终端节点 EC2。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](#)。

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 (北加利福尼亚)	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	
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	

区域名称	区域	端点	协议	
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 (马来西亚)	ap-southeast-5	ec2.ap-southeast-5.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 (新西兰)	ap-southeast-6	ec2.ap-southeast-6.amazonaws.com	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 (台北)	ap-east-2	ec2.ap-east-2.amazonaws.com	HTTPS	
亚太地区 (泰国)	ap-southeast-7	ec2.ap-southeast-7.amazonaws.com	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	

区域名称	区域	端点	协议	
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	
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	
墨西哥 (中部)	mx-central-1	ec2.mx-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	

区域名称	区域	端点	协议	
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	
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	

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

以下示例显示在使用 AWS CLI 时，如何为 us-east-2 区域指定端点。

- 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

以下示例显示在使用 AWS SDK for Java 2.x 时，如何为 us-east-2 区域指定端点。

- 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

以下示例显示在使用 AWS SDK for Go 时，如何为 us-east-1 区域指定端点。

- 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 Amazon EC2 API 的系统的分布式特性，Amazon 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给您随后运行的所有命令。在执行紧随上一个 API 命令的 API 命令时，应记住这一点。

最终一致性会影响您管理资源的方式。例如，如果您运行命令创建了某个资源，这个资源最终会对其他命令变得可见。这意味着，如果您立刻运行命令去修改或查询刚创建的资源，由于其 ID 可能尚未在系统中传播完成，您可能会收到资源不存在的错误。

要管理最终一致性，您可以执行以下操作：

- 在运行修改命令之前，先确认资源的当前状态。使用指数回退算法运行相应的 Describe 命令，来确保有足够的时间让前一个命令传播遍整个系统。为此，请重复运行该 Describe 命令，从几秒钟的等待时间开始，然后逐渐增加到几分钟的等待时间。
- 增加后续命令之间的等待时间，即使 Describe 命令返回准确的响应，也是如此。应用指数退避算法，从几秒钟的等待时间开始，然后逐渐增加到几分钟的等待时间。

最终一致性错误示例

以下是一些由于最终一致性而可能遇到的错误代码示例。

- InvalidInstanceID.NotFound

如果您成功执行了 `RunInstances` 命令，然后立刻使用 `RunInstances` 返回的实例 ID 执行另一个命令，可能会返回 `InvalidInstanceID.NotFound` 错误。这并不代表该实例不存在。

以下是可能受影响的一些具体命令：

- `DescribeInstances`：要确认实例的实际状态，请使用指数回退算法运行此命令。
- `TerminateInstances`：要确认实例状态，请先使用指数回退算法运行 `DescribeInstances` 命令。

Important

如果您在运行 `TerminateInstances` 之后收到 `InvalidInstanceID.NotFound` 错误，这并不代表实例已经或即将被终止。实例仍有可能正在运行。因此，您应当首先使用 `DescribeInstances` 来确认实例状态。

- `InvalidGroup.NotFound`

如果您成功执行了 `CreateSecurityGroup` 命令，然后立即使用 `CreateSecurityGroup` 返回的安全组 ID 执行另一个命令，可能会返回 `InvalidGroup.NotFound` 错误。要确认安全组的状态，请使用指数回退算法运行 `DescribeSecurityGroups` 命令。

- `InstanceLimitExceeded`

您请求的实例数量超过了当前对指定实例类型的配额上限。如果您在短时间内频繁创建和终止实例，可能会出乎意料地触及此配额限制，因为即使实例已被终止，它们在一段时间内仍会计入您的实例配额。

Ensuring idempotency in Amazon EC2 API requests

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。这意味着您可能会创建超过预期数量的资源。

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。不过，结果中可能包含更新后的信息，例如当前的创建状态。

目录

- [Amazon EC2 中的幂等性](#)

- [RunInstances 的幂等性](#)
- [示例](#)
- [针对幂等请求的重试建议](#)

Amazon EC2 中的幂等性

以下 API 操作默认具备幂等性，无需额外配置。对应的 AWS CLI 命令也默认支持幂等性。

默认具备幂等性的操作

- AssociateAddress
- CreateVpnConnection
- DisassociateAddress
- ReplaceNetworkAclAssociation
- TerminateInstances

以下 API 操作通过客户端令牌来支持幂等性（可选）。对应的 AWS CLI 命令同样支持通过客户端令牌实现幂等性。客户端令牌是一个由最多 64 个 ASCII 字符组成、区分大小写的唯一字符串。要让这些操作的 API 请求具备幂等性，请在请求中指定一个客户端令牌。您不应将同一个客户端令牌用于其他 API 请求。如果您使用相同的客户端令牌并保持所有参数一致来重试一个已成功完成的请求，该重试会直接成功，而不会执行任何额外操作。如果您在重试成功的请求时使用了相同的客户端令牌，但一个或多个参数发生变化（区域或可用区除外），重试将失败，并返回 `IdempotentParameterMismatch` 错误。

使用客户端令牌实现幂等性的操作

- AllocateHosts
- AllocateIamPoolCidr
- AssociateClientVpnTargetNetwork
- AssociateIamResourceDiscovery
- AttachVerifiedAccessTrustProvider
- AuthorizeClientVpnIngress
- CopyFpgaImage
- CopyImage

- `CreateCapacityReservation`
- `CreateCapacityReservationFleet`
- `CreateClientVpnEndpoint`
- `CreateClientVpnRoute`
- `CreateEgressOnlyInternetGateway`
- `CreateFleet`
- `CreateFlowLogs`
- `CreateFpgaImage`
- `CreateInstanceConnectEndpoint`
- `CreateIam`
- `CreateIamPool`
- `CreateIamResourceDiscovery`
- `CreateIamScope`
- `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

幂等性类型

- 区域级 – 请求在每个区域内具备幂等性。然而，您可以在其他区域中使用同一个请求（包括相同的客户端令牌）。
- 可用区级 – 请求在一个区域内的每个可用区中具备幂等性。例如，如果您在同一区域内的两次 `AllocateHosts` 调用中使用相同的客户端令牌，只要它们的 `AvailabilityZone` 参数值不同，这两个调用都会成功。

RunInstances 的幂等性

[RunInstances](#) API 操作同时采用区域级和可用区级幂等性。

具体采用哪种幂等性，取决于您在 `RunInstances` API 请求中如何指定可用区。在以下情况中，请求会使用可用区级幂等性：

- 您在 `Placement` 数据类型中，通过 `AvailabilityZone` 参数显式指定了可用区
- 您使用 `SubnetId` 参数隐式指定了可用区

您既没有显式也没有隐式指定可用区，则请求会使用区域级幂等性。

可用区级幂等性

可用区级幂等性确保 `RunInstances` API 请求在同一区域内的每个可用区中都是幂等的。这保证了在同一区域内，使用相同客户端令牌的请求在每个可用区中只能成功执行一次。然而，您仍然可以使用同一个客户端令牌在该区域的其他可用区中启动实例。

例如，如果您发送一个在 `us-east-1a` 可用区启动实例的幂等请求，然后在 `us-east-1b` 可用区再次使用相同的客户端令牌发起请求，我们会在这些可用区中分别启动实例。如果后续重试中的一个或多个参数发生变化，那么在这些可用区内使用相同客户端令牌进行的重试要么会成功返回但不执行任何操作，要么会因参数不匹配而报告 `IdempotentParameterMismatch` 错误。

区域级幂等性

区域级幂等性确保 RunInstances API 请求在同一区域内是幂等的。这保证了在同一区域内，使用相同客户端令牌的请求只能成功执行一次。然而，您可以使用完全相同的请求（包括相同的客户端令牌）在其他区域中启动实例。

例如，如果您发送一个在 us-east-1 区域启动实例的幂等请求，然后在 eu-west-1 区域中再次使用相同的客户端令牌发起请求，我们会在两个区域中分别启动实例。如果后续重试中的一个或多个参数发生变化，那么在這些区域内使用相同客户端令牌进行的重试要么会成功返回但不执行任何操作，要么会因参数不匹配而报告 IdempotentParameterMismatch 错误。

Tip

如果请求的区域中有某个可用区不可用，使用区域级幂等性的 RunInstances 请求可能会失败。为了充分利用 AWS 基础设施提供的可用区功能，我们建议在启动实例时使用可用区级幂等性。即使请求的区域中另一个可用区不可用，只要目标可用区可用，使用可用区级幂等性的 RunInstances 请求仍然会成功。

示例

AWS CLI 命令示例

要使 AWS CLI 命令具备幂等性，请添加 `--client-token` 选项。

示例 1：幂等性

以下 [allocate-hosts](#) 命令通过包含客户端令牌而具备幂等性。

```
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：run-instances 的区域级幂等性

以下 [run-instances](#) 命令通过包含客户端令牌而使用区域级幂等性，因为它既没有显式也没有隐式指定可用区。

```
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：run-instances 的可用区级幂等性

以下 [run-instances](#) 命令通过包含客户端令牌并显式指定可用区而使用可用区级幂等性。

```
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 请求具有幂等性，请添加 ClientToken 参数。

示例 1：幂等性

以下 [AllocateHosts](#) API 请求通过包含客户端令牌而具备幂等性。

```
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 的区域级幂等性

以下 [RunInstances](#) API 请求通过包含客户端令牌而使用区域级幂等性，因为它既没有显式也没有隐式指定可用区。

```
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 的可用区级幂等性

以下 [RunInstances](#) API 请求通过包含客户端令牌并显式指定可用区而使用可用区级幂等性。

```
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
```

针对幂等请求的重试建议

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	评论
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 系列响应代码 (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 系列响应代码 (服务器端错误)	重试	该错误是由 AWS 服务器端问题引起，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 API 的请求限制 EC2

Amazon EC2 会按区域限制每个 AWS 账户的 EC2 API 请求。我们这样做是为了帮助提高服务的性能，并确保所有亚马逊 EC2 买家都能公平使用。限制可确保对 Amazon EC2 API 的请求不超过允许的最大 API 请求限制。无论 API 请求来自以下哪些来源，都会受到请求限制的约束：

- 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 节流限制，则会收到 `RequestLimitExceeded` 错误代码。

内容

- [如何应用节流](#)
- [请求令牌存储桶的容量与重填速率](#)
- [资源令牌存储桶容量和重填速率](#)
- [监控 API 节流](#)
- [重试和指数回退](#)
- [请求提高限制](#)

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。存储桶中令牌的数量表示您在任意一秒的节流限制。

Amazon EC2 实施了两种类型的 API 限制：

API 节流类型

- [请求速率限制](#)
- [资源速率限制](#)

请求速率限制

在请求速率限制中，每个 API 都会被单独评估，您的节流限制按每个 API 的请求数量分别计算。您每发起一次请求，就会从该 API 的存储桶中移除一个令牌。例如，非变异 API 操作的 `DescribeHosts` 的令牌存储桶容量为 100 个令牌。您在一秒内最多可以发起 100 次 `DescribeHosts` 请求。如果在某一秒内超过 100 次请求，您会在该 API 上被节流，剩余请求会失败；但其他 API 的请求不受影响。

存储桶会以设定的速率自动填充。如果存储桶的容量低于其最大容量，则每秒都会向其添加一定数量的令牌，直到其达到最大容量。如果重填令牌到达时存储桶已满，额外的令牌会被丢弃。存储桶中的令牌数量不能超过其最大数量。例如，非变异 API 操作的 `DescribeHosts` 的令牌存储桶容量为 100 个令

牌，重填速率为每秒 20 个令牌。如果您在一秒钟内发起 100 次 DescribeHosts 请求，存储桶中的令牌将被消耗至零 (0)。之后，存储桶会以每秒 20 个的速度重填令牌，直到重新达到 100 个的最大容量。这意味着，如果期间没有请求，一个空存储桶需要 5 秒才能重新装满。

您无需等存储桶完全填满才能继续发起 API 请求。您可以使用刚添加进存储桶的重填令牌。如果您立即使用重填令牌，存储桶就不会达到最大容量。例如，非变异 API 操作的 DescribeHosts 的令牌存储桶容量为 100 个令牌，重填速率为每秒 20 个令牌。如果您在一秒内用掉存储桶中的全部 100 个令牌，仍可以随后每秒发起 20 次请求，使用的就是不断重填到存储桶中的令牌。只有当您每秒发起的请求少于 20 次时，存储桶才有机会重新填满。

有关更多信息，请参阅 [请求令牌存储桶的容量与重填速率](#)。

资源速率限制

某些 API 操作（例如 RunInstances 和 TerminateInstances，如下表所示），除了请求速率限制外，还使用资源速率限制。这些 API 会有一个独立的资源令牌存储桶，根据请求影响的资源数量来消耗令牌。与请求令牌存储桶类似，资源令牌存储桶也有一个最大容量用于支持突发请求，同时具备重填速率，使您能够在需要时持续保持稳定的请求速率。如果您超出了某个 API 的资源存储桶限制（包括当存储桶尚未重填到足以支持下一次 API 请求时的情况），即使整体 API 节流限制未达到，您仍会被限制执行该操作。

例如，RunInstances 的资源令牌存储桶容量为 1000 个令牌，重填速率为每秒 2 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，例如一次请求启动 1000 个，或者分四次请求，每次启动 250 个实例。在资源令牌存储桶耗尽后，您每秒最多只能启动两个实例，可以是一次请求启动两个，也可以是两次请求各启动一个。

有关更多信息，请参阅 [资源令牌存储桶容量和重填速率](#)。

请求令牌存储桶的容量与重填速率

为实现请求速率限制，API 操作分为以下几类：

- 非变异操作 – 用于检索资源相关数据的 API 操作。此类别通常包括所有 Describe*、List*、Search* 和 Get* API 操作，例如 DescribeRouteTables、SearchTransitGatewayRoutes 和 GetIpamPoolCidrs。这些 API 操作通常具有最高的 API 节流限制。
- 未筛选、未分页的非变异操作 – 非变异 API 操作中的一个特定子集，如果请求未指定[分页](#)或[筛选条件](#)，则会从一个较小的令牌存储桶中扣除令牌。建议您在调用这些操作时使用分页和筛选，以便令牌从标准（容量更大）的存储桶中扣除。

- 变异操作 – 用于创建、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异操作的 API 操作，例如 `AllocateHosts`、`ModifyHosts` 和 `CreateCapacityReservation`。这些操作的节流限制低于非变异 API 操作。
- 资源密集型操作 – 完成时间最长、消耗资源最多的变异类 API 操作。这些操作的节流限制比变异操作更低。它们的节流是独立于其他变异操作的。
- 控制台非变更操作 — 从 Amazon 控制台请求的非变更 API 操作。EC2 这些 API 操作的节流是独立于其他非变异 API 操作的。
- 未分类操作 – 这些 API 操作会拥有独立的令牌存储桶容量和重填速率，即便从定义上它们原本属于上述某一类别。

API 操作类别	操作	存储桶最大容量	存储桶重填速率
非变异操作	未包含在其他类别中的 <code>Describe*</code> 、 <code>List*</code> 、 <code>Search*</code> 和 <code>Get*</code> API 操作。	100	20
未筛选、未分页的非变异操作	• <code>DescribeInstances</code> • <code>DescribeInstanceStatus</code> • <code>DescribeNetworkInterfaces</code> • <code>DescribeSecurityGroups</code> • <code>DescribeSnapshots</code>	50	10

API 操作类别	操作	存储桶最大容量	存储桶重填速率
	- DescribeSpotInstanceRequests - DescribeVolumes		
变异操作	所有不是资源密集型操作或未分类操作的变异 API 操作。	50	5

API 操作类别	操作	存储桶最大容量	存储桶重填速率
资源密集型操作	- AcceptVpcPeeringConnection - AuthorizeSecurityGroupIngress - CancelSpotInstanceRequests - CreateKeyPair - CreateVpcPeeringConnection - DeleteVpcPeeringConnection - RejectVpcPeeringConnection - RevokeSecurityGroupIngress - RequestSpotInstances	50	5

API 操作类别	操作	存储桶最大容量	存储桶重填速率
控制台非变异操作	Describe* 、List*Search*、 和 Get* API 操作，由 亚马逊 EC2 控制台调 用，但未包含在其他 类别中。	100	10

未分类操作	存储桶最大容量	存储桶重填速率
AcceptVpcEndpointConnections	10	1
AdvertiseByoipCidr	1	0.1
AssignIpv6Addresses	100	5
AssignPrivateIpAddresses	100	5
AssignPrivateNatGatewayAddress	10	1
AssociateCapacityReservationBillingOwner	1	0.5
AssociateEnclaveCertificateIamRole	10	1
AssociateIamInstanceProfile	100	5
AssociateNatGatewayAddress	10	1
AttachVerifiedAccessTrustProvider	10	2
AuthorizeClientVpnIngress	5	2
CancelDeclarativePoliciesReport	1	1
CopyImage	100	1

未分类操作	存储桶最大容量	存储桶重填速率
CreateClientVpnRoute	5	2
CreateCoipCidr	5	1
CreateCoipPool	5	1
CreateDefaultSubnet	1	1
CreateDefaultVpc	1	1
CreateLaunchTemplateVersion	100	5
CreateNatGateway	10	1
CreateNetworkInterface	100	5
CreateRestoreImageTask	50	0.1
CreateSnapshot	100	5
CreateSnapshots	100	5
CreateSpotDatafeedSubscription	50	3
CreateStoreImageTask	50	0.1
CreateSubnetCidrReservation	5	1
CreateTags	100	10
CreateVerifiedAccessEndpoint	20	4
CreateVerifiedAccessGroup	10	2
CreateVerifiedAccessInstance	10	2
CreateVerifiedAccessTrustProvider	10	2
CreateVolume	100	5

未分类操作	存储桶最大容量	存储桶重填速率
CreateVpcEndpoint	4	0.3
CreateVpcEndpointServiceConfiguration	10	1
DeleteClientVpnRoute	5	2
DeleteCoipCidr	5	1
DeleteCoipPool	5	1
DeleteCoipPoolPermission	5	1
DeleteNatGateway	10	1
DeleteNetworkInterface	100	5
DeleteSnapshot	100	5
DeleteSpotDatafeedSubscription	50	3
DeleteSubnetCidrReservation	5	1
DeleteQueuedReservedInstances	5	5
DeleteTags	100	10
DeleteVerifiedAccessEndpoint	20	4
DeleteVerifiedAccessGroup	10	2
DeleteVerifiedAccessInstance	10	2
DeleteVerifiedAccessTrustProvider	10	2
DeleteVolume	100	5
DeleteVpcEndpoints	4	0.3
DeleteVpcEndpointServiceConfigurations	10	1

未分类操作	存储桶最大容量	存储桶重填速率
DeprovisionByoipCidr	1	0.1
DeregisterImage	100	5
DescribeAggregateIdFormat	10	10
DescribeByoipCidrs	1	0.5
DescribeCapacityBlockExtensionOfferings	10	0.15
DescribeCapacityBlockOfferings	10	0.15
DescribeDeclarativePoliciesReports	5	5
DescribeHostReservations	5	2
DescribeHostReservationOfferings	5	2
DescribeIdentityIdFormat	10	10
DescribeIdFormat	10	10
DescribeInstanceTopology	1	1
DescribeMovingAddresses	1	1
DescribePrincipalIdFormat	10	10
DescribeReservedInstancesOfferings	10	10
DescribeSecurityGroupReferences	20	5
DescribeSpotDatafeedSubscription	100	13
DescribeSpotFleetInstances	100	5
DescribeSpotFleetRequestHistory	100	5
DescribeSpotFleetRequests	50	3

未分类操作	存储桶最大容量	存储桶重填速率
DescribeStaleSecurityGroups	20	5
DescribeStoreImageTasks	50	0.5
DescribeVerifiedAccessInstanceLoggingConfigurations	10	2
DetachVerifiedAccessTrustProvider	10	2
DisableFastLaunch	5	2
DisableImageBlockPublicAccess	1	0.1
DisableSnapshotBlockPublicAccess	1	0.1
DisassociateCapacityReservationBillingOwner	1	0.5
DisassociateEnclaveCertificateIamRole	10	1
DisassociateIamInstanceProfile	100	5
DisassociateNatGatewayAddress	10	1
EnableFastLaunch	5	2
EnableImageBlockPublicAccess	1	0.1
EnableSnapshotBlockPublicAccess	1	0.1
GetAssociatedEnclaveCertificateIamRoles	10	1
GetDeclarativePoliciesReportSummary	5	5
GetHostReservationPurchasePreview	5	2
ModifyImageAttribute	100	5

未分类操作	存储桶最大容量	存储桶重填速率
ModifyInstanceMetadataDefaults	2	2
ModifyInstanceMetadataOptions	100	5
ModifyLaunchTemplate	100	5
ModifyNetworkInterfaceAttribute	100	5
ModifySnapshotAttribute	100	5
ModifyVerifiedAccessEndpoint	20	4
ModifyVerifiedAccessEndpointPolicy	20	4
ModifyVerifiedAccessGroup	10	2
ModifyVerifiedAccessGroupPolicy	20	4
ModifyVerifiedAccessInstance	10	2
ModifyVerifiedAccessInstanceLoggingConfiguration	10	2
ModifyVerifiedAccessTrustProvider	10	2
ModifyVpcEndpoint	4	0.3
ModifyVpcEndpointServiceConfiguration	10	1
MoveAddressToVpc	1	1
ProvisionByoipCidr	1	0.1
PurchaseCapacityBlock	10	0.15
PurchaseCapacityBlockExtension	10	0.15
PurchaseHostReservation	5	2
PurchaseReservedInstancesOffering	5	5

未分类操作	存储桶最大容量	存储桶重填速率
RejectVpcEndpointConnections	10	1
RestoreAddressToClassic	1	1
RevokeClientVpnIngress	5	2
RunInstances	5	2
StartDeclarativePoliciesReport	1	1
StartInstances	5	2
TerminateInstances	100	5
UnassignPrivateIpAddresses	100	5
UnassignPrivateNatGatewayAddress	10	1
WithdrawByoipCidr	1	0.1

资源令牌存储桶容量和重填速率

下表列出了使用资源速率限制的 API 操作所对应的资源令牌存储桶容量及其重填速率。

API 操作	存储桶最大容量	存储桶重填速率
RunInstances	1000	2
TerminateInstances	1000	20
StartInstances	1000	2
StopInstances	1000	20

监控 API 节流

您可以使用亚马逊 CloudWatch 来监控您的亚马逊 EC2 API 请求，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 节流限制时向您发出警告。有关更多信息，请参阅 [使用亚马逊监控亚马逊 EC2 API 请求 CloudWatch](#)。

重试和指数回退

您的应用程序可能需要重试某个 API 请求。例如：

- 检查资源状态是否更新
- 枚举大量资源（例如，您的所有卷）
- 在请求因服务端错误（5xx）或节流错误而失败时执行重试

但如果是客户端错误（4xx），您必须先修正请求中的问题，再进行重试。

资源状态更改

在开始轮询检查状态更新之前，应先预留足够时间让请求可能完成。例如，等待几分钟，然后再检查实例是否处于活动状态。当开始轮询时，应在连续请求之间设置合理的睡眠间隔，以降低 API 请求速率。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 [EC2 使用自动化 Amazon EventBridge](#)。

重试

当您需要轮询或重试 API 请求时，我们建议使用指数回退算法来计算请求之间的睡眠间隔。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。您还可以使用抖动（随机延迟）来避免请求持续冲突。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《工具参考指南》AWS SDKs 和《工具参考指南》中的[重试行为](#)。

请求提高限制

您可以请求增加您的 AWS 账户的 API 节流限制。

建议

- 单次请求的限制最多不应超过您当前限制的三倍。
- 优先请求提升存储桶的重填速率，再考虑提升存储桶的最大容量。
- 如果您请求的重填速率会超过当前的存储桶最大容量，则应同时请求提升最大容量。
- 提供所有需要提升限制的 API 操作。限制是应用在单个 API 操作上的，而不是应用在整个 API 操作类别上。
- 以下 API 操作拥有请求速率限制和资源速率限制：RunInstances、StartInstances、StopInstances 和 TerminateInstances。务必说明您希望提升的是哪一种限制。

请求访问此功能

1. 打开 [AWS Support 中心](#)。
2. 选择创建工单。
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，复制以下模板并提供所需信息。

```
Please increase the API throttling limits for my account.  
Related page: https://docs.aws.amazon.com/ec2/latest/devguide/ec2-api-throttling.html  
Description: Brief notes about your use case. If available, include the IDs of a few Amazon EC2 requests that were throttled.  
Time window: One-hour window when peak throttling or usage occurred.  
region_1 request rate increases:  
    action: new_bucket_maximum_capacity  
    action: new_bucket_refill_rate  
    action: new_bucket_maximum_capacity|new_bucket_refill_rate  
region_1 resource rate increases:  
    action: new_bucket_maximum_capacity  
    action: new_bucket_refill_rate  
    action: new_bucket_maximum_capacity|new_bucket_refill_rate
```

```
region_2 request rate increases:
  action: new_bucket_maximum_capacity
  action: new_bucket_refill_rate
  action: new_bucket_maximum_capacity|new_bucket_refill_rate
region_2 resource rate increases:
  action: new_bucket_maximum_capacity
  action: new_bucket_refill_rate
  action: new_bucket_maximum_capacity|new_bucket_refill_rate
```

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首选联系方式。
11. 选择提交。

Amazon EC2 API 中的分页

我们建议在调用可能返回大量结果的描述类操作（例如 `DescribeInstances`）时使用分页。使用分页可以限制每次描述调用返回的项目数量，并控制调用返回所需的时间。如果您拥有大量资源，未分页的调用可能会被限制速率，甚至可能超时。因此，相比未分页调用，分页调用通常更稳定成功，从而带来更好的整体延迟表现。

有关更多信息，请参阅《Amazon EC2 API 参考》中的[分页](#)。

最佳实践

在可能的情况下，在描述调用中指定资源 ID 列表。这是描述大量资源的最快捷方式。请注意，单次调用中不应指定超过 1000 个 ID。示例如下：

```
private List<Reservation> describeMyInstances(List<String> ids){
    if (ids == null || ids.isEmpty()) {
        return ImmutableList.of();
    }

    final DescribeInstancesRequest request = new DescribeInstancesRequest()
        .withInstanceIds(ids);

    return ec2.describeInstances(request).getReservations();
}
```

如果您无法在描述调用中指定资源 ID，我们强烈建议使用分页。示例如下：

```
private List<Reservation> describeMyInstances(final Collection<Filter> filters){
    final DescribeInstancesRequest request = new DescribeInstancesRequest()
        .withFilters(filters)
        .withMaxResults(1000);

    List<Reservation> reservations = new ArrayList<>();
    String nextToken = null;
    do {
        request.setNextToken(nextToken);
        final DescribeInstancesResult response = ec2.describeInstances(request);
        reservations.addAll(response.getReservations());
        nextToken = response.getNextToken();
    } while (nextToken != null);

    return reservations;
}
```

如果需要重试分页调用，请使用[带抖动的指数退避算法](#)。

常见问题

以下是一些无意间发起未分页调用的代码示例。

Example问题示例：传递一个空的资源 ID 列表

下面的代码使用了 ID 列表。然而，如果该列表为空，最终会导致发起未分页的调用。

```
private List<Reservation> describeMyInstances(List<String> ids){
    final DescribeInstancesRequest request = new DescribeInstancesRequest()
        .withInstanceIds(ids);

    return ec2.describeInstances(request).getReservations();
}
```

要修正此问题，请确保在执行描述调用之前列表不为空。

```
private List<Reservation> describeMyInstances(List<String> ids){
    if (ids == null || ids.isEmpty()) {
        return ImmutableList.of();
        // OR
        return Lists.newArrayList();
        // OR
    }
}
```

```
        return new ArrayList<>();
    }

    final DescribeInstancesRequest request = new DescribeInstancesRequest()
        .withInstanceIds(ids);

    return ec2.describeInstances(request).getReservations();
}
```

Example问题示例：未设置 MaxResults

以下代码虽然检查并使用了 nextToken，但并未设置 MaxResults。

```
private List<Reservation> describeMyInstances(final Collection<Filter> filters){
    final DescribeInstancesRequest request = new DescribeInstancesRequest()
        .withFilters(filters);

    List<Reservation> reservations = new ArrayList<>();
    String nextToken = null;
    do {
        request.setNextToken(nextToken);
        final DescribeInstancesResult response = ec2.describeInstances(request);
        reservations.addAll(response.getReservations());
        nextToken = response.getNextToken();
    } while (nextToken != null);

    return reservations;
}
```

要解决此问题，请按以下方式添加 withMaxResults。

```
private List<Reservation> describeMyInstances(final Collection<Filter> filters){
    final DescribeInstancesRequest request = new DescribeInstancesRequest()
        .withFilters(filters)
        .withMaxResults(1000);

    List<Reservation> reservations = new ArrayList<>();
    String nextToken = null;
    do {
        request.setNextToken(nextToken);
        final DescribeInstancesResult response = ec2.describeInstances(request);
        reservations.addAll(response.getReservations());
        nextToken = response.getNextToken();
    } while (nextToken != null);

    return reservations;
}
```

```
    } while (nextToken != null);  
  
    return reservations;  
}
```

使用 AWS CLI 创建 Amazon EC2 资源

您可以在命令行 Shell 中使用 AWS Command Line Interface (AWS CLI) 来创建和管理 Amazon EC2 资源。通过 AWS CLI 可直接访问包括 Amazon EC2 在内的 AWS 服务的 API。

有关 Amazon EC2 命令的语法和示例，请参阅《AWS CLI 命令参考》中的 [ec2](#)。您也可以在 GitHub 的 [aws-cli/awsccli/examples/ec2](#) 中找到这些示例。

了解有关 AWS CLI 的更多信息

要了解有关 AWS CLI 的更多信息，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 AWS CloudFormation 创建 Amazon EC2 资源

Amazon EC2 与 AWS CloudFormation 集成，后者是一项服务，可协助您建模并设置 AWS 资源，这样您只需花较少的时间来创建和管理资源与基础设施。您可以创建一个描述所需 AWS 资源（例如实例和子网）的模板，CloudFormation 将为您预调配和配置这些资源。

使用 CloudFormation 时，您可以重复使用模板，以一致的方式多次设置 Amazon EC2 资源。描述您的资源一次，然后在多个 AWS 账户和区域中反复预置相同的资源。

Amazon EC2 和 CloudFormation 模板

要为 Amazon EC2 和相关服务预调配和配置资源，您必须了解 [CloudFormation 模板](#)。模板是 JSON 或 YAML 格式的文本文件。这些模板可描述您要在 CloudFormation 堆栈中预调配的资源。如果您不熟悉 JSON 或 YAML，可以在 CloudFormation Designer 的帮助下开始使用 CloudFormation 模板。有关更多信息，请参阅《AWS CloudFormation 用户指南》中的[什么是 CloudFormation Designer？](#)。

Amazon EC2 的资源

计算资源

- [AWS::EC2::CapacityReservation](#)
- [AWS::EC2::CapacityReservationFleet](#)
- [AWS::EC2::EC2Fleet](#)
- [AWS::EC2::EC2Fleet](#)
- [AWS::EC2::Host](#)
- [AWS::EC2::Instance](#)
- [AWS::EC2::InstanceConnectEndpoint](#)
- [AWS::EC2::LaunchTemplate](#)
- [AWS::EC2::PlacementGroup](#)
- [AWS::EC2::SpotFleet](#)

联网资源

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

- [AWS::EC2::ClientVpnAuthorizationRule](#)
- [AWS::EC2::ClientVpnEndpoint](#)
- [AWS::EC2::ClientVpnRoute](#)
- [AWS::EC2::ClientVpnTargetNetworkAssociation](#)
- [AWS::EC2::CustomerGateway](#)
- [AWS::EC2::DHCPOptions](#)
- [AWS::EC2::EgressOnlyInternetGateway](#)
- [AWS::EC2::EIP](#)
- [AWS::EC2::EIPAssociation](#)
- [AWS::EC2::FlowLog](#)
- [AWS::EC2::GatewayRouteTableAssociation](#)
- [AWS::EC2::InternetGateway](#)
- [AWS::EC2::IPAM](#)
- [AWS::EC2::IPAMAllocation](#)
- [AWS::EC2::IPAMPool](#)
- [AWS::EC2::IPAMPoolCidr](#)
- [AWS::EC2::IPAMResourceDiscovery](#)
- [AWS::EC2::IPAMResourceDiscoveryAssociation](#)
- [AWS::EC2::IPAMScope](#)
- [AWS::EC2::LocalGatewayRoute](#)
- [AWS::EC2::LocalGatewayRouteTable](#)
- [AWS::EC2::LocalGatewayRouteTableVirtualInterfaceGroupAssociation](#)
- [AWS::EC2::LocalGatewayRouteTableVPCAssociation](#)
- [AWS::EC2::NatGateway](#)
- [AWS::EC2::NetworkInterface](#)
- [AWS::EC2::NetworkInsightsAccessScope](#)
- [AWS::EC2::NetworkInsightsAccessScopeAnalysis](#)
- [AWS::EC2::NetworkInsightsAnalysis](#)
- [AWS::EC2::NetworkInsightsPath](#)
- [AWS::EC2::NetworkInterfaceAttachment](#)

- [AWS::EC2::NetworkInterfacePermission](#)
- [AWS::EC2::NetworkPerformanceMetricSubscription](#)
- [AWS::EC2::PrefixList](#)
- [AWS::EC2::Route](#)
- [AWS::EC2::RouteTable](#)
- [AWS::EC2::Subnet](#)
- [AWS::EC2::SubnetCidrBlock](#)
- [AWS::EC2::SubnetNetworkAclAssociation](#)
- [AWS::EC2::SubnetRouteTableAssociation](#)
- [AWS::EC2::TrafficMirrorFilter](#)
- [AWS::EC2::TrafficMirrorFilterRule](#)
- [AWS::EC2::TrafficMirrorSession](#)
- [AWS::EC2::TrafficMirrorTarget](#)
- [AWS::EC2::TransitGateway](#)
- [AWS::EC2::TransitGatewayAttachment](#)
- [AWS::EC2::TransitGatewayConnect](#)
- [AWS::EC2::TransitGatewayMulticastDomain](#)
- [AWS::EC2::TransitGatewayMulticastDomainAssociation](#)
- [AWS::EC2::TransitGatewayMulticastGroupMember](#)
- [AWS::EC2::TransitGatewayMulticastGroupSource](#)
- [AWS::EC2::TransitGatewayPeeringAttachment](#)
- [AWS::EC2::TransitGatewayRoute](#)
- [AWS::EC2::TransitGatewayRouteTable](#)
- [AWS::EC2::TransitGatewayRouteTableAssociation](#)
- [AWS::EC2::TransitGatewayRouteTablePropagation](#)
- [AWS::EC2::TransitGatewayVpcAttachment](#)
- [AWS::EC2::VPC](#)
- [AWS::EC2::VPCcidrBlock](#)
- [AWS::EC2::VPCDHCPOptionsAssociation](#)
- [AWS::EC2::VPCEndpoint](#)

- [AWS::EC2::VPCEndpointConnectionNotification](#)
- [AWS::EC2::VPCEndpointService](#)
- [AWS::EC2::VPCEndpointServicePermissions](#)
- [AWS::EC2::VPCGatewayAttachment](#)
- [AWS::EC2::VPCPeeringConnection](#)
- [AWS::EC2::VPNConnection](#)
- [AWS::EC2::VPNConnectionRoute](#)
- [AWS::EC2::VPNGateway](#)
- [AWS::EC2::VPNGatewayRoutePropagation](#)

安全性资源

- [AWS::EC2::KeyPair](#)
- [AWS::EC2::NetworkAcl](#)
- [AWS::EC2::NetworkAclEntry](#)
- [AWS::EC2::SecurityGroup](#)
- [AWS::EC2::SecurityGroupEgress](#)
- [AWS::EC2::SecurityGroupIngress](#)
- [AWS::EC2::VerifiedAccessEndpoint](#)
- [AWS::EC2::VerifiedAccessGroup](#)
- [AWS::EC2::VerifiedAccessInstance](#)
- [AWS::EC2::VerifiedAccessTrustProvider](#)

存储资源

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

了解有关 CloudFormation 的更多信息

要了解有关 CloudFormation 的更多信息，请参阅以下资源：

- [AWS CloudFormation](#)
- [AWS CloudFormation 《用户指南》](#)

使用 AWS SDK 创建 Amazon EC2 资源

AWS 为多种主流编程语言提供软件开发工具包 (SDK)。SDK 通过提供以下功能来提升开发效率：

- 可直接集成到应用程序中的预构建组件和库
- 面向特定语言的工具，如编译器和调试器
- 服务请求的加密签名
- 请求重试机制
- 错误响应处理

Amazon EC2 API 的代码示例

AWS 提供的代码示例可帮助您了解如何使用 API 并完成特定任务。有关 Amazon EC2 API 的示例，请参阅 [Amazon EC2 的代码示例](#)。如需更多示例，请参阅[查找 AWS SDK 的代码示例](#)或 GitHub 上的[aws-doc-sdk-examples](#)。

详细了解 AWS SDK

要了解有关 AWS SDK 的更多信息，请参阅以下资源：

- [AWS SDK 和工具参考指南](#)
- [AWS 的构建工具](#)
- [什么是 SDK？](#)

Amazon EC2 的低级别 API

Amazon EC2 的低级别 API 是 Amazon EC2 的协议级接口。使用低级别 API 时，您必须正确格式化每一个 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《Amazon EC2 API 参考》中的[向 Amazon EC2 API 发起请求](#)。或者，您也可以使用 AWS SDK，由其代表您构建并签署请求。有关更多信息，请参阅[使用 AWS 软件开发工具包](#)。

Amazon EC2 API 由多个服务的操作与数据类型组成。要查看各服务的操作，请参阅《Amazon EC2 API 参考》中的以下页面。

- [AWS Client VPN 操作](#)
- [Amazon EBS 操作](#)
- [Amazon EC2 操作](#)
- [AWS Network Manager 操作](#)
- [AWS Nitro Enclaves 操作](#)
- [AWS Outposts 操作](#)
- [AWS PrivateLink 操作](#)
- [Recycle Bin 操作](#)
- [AWS Site-to-Site VPNActions](#)
- [AWS Transit Gateway 操作](#)
- [AWS Verified Access 操作](#)
- [VM Import/Export 操作](#)
- [Amazon VPC 操作](#)
- [Amazon VPC IPAM 操作](#)
- [AWS Wavelength 操作](#)

使用 Console-to-Code 为您的 EC2 控制台操作生成代码

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 Q 开发者版的一项功能，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，包括默认设置和兼容参数。随后，其利用生成式人工智能，以您首选的基础设施即代码 (IaC) 格式，为您要执行的操作提供代码建议。由于控制台工作流程可确保您指定的参数值同时有效，因此，您使用 Console-to-Code 生成的代码具有兼容的参数值。您可以将此代码用作一个起点，然后对其进行自定义，以使它可用于您的特定使用案例的生产。

例如，使用 Console-to-Code，可以记录启动一个 Amazon EC2 实例的操作，并选择以 CloudFormation JSON 格式生成代码。然后，您可以复制该代码并对其进行自定义，以便在 CloudFormation 模板中使用。

Console-to-Code 目前能够以下列语言和格式生成基础设施即代码 (IaC)：

- CDK Java
- CDK Python
- CDK TypeScript
- CloudFormation JSON
- CloudFormation YAML

有关如何使用 Console-to-Code 的更多信息和说明，请参阅《Amazon Q Developer User Guide》中的 [Automating AWS services with Amazon Q Developer Console-to-Code](#)。

EC2 使用 Amazon 的代码示例 AWS SDKs

以下代码示例展示了如何将 Amazon EC2 与 AWS 软件开发套件 (SDK) 一起使用。

基本功能是向您展示如何在服务中执行基本操作的代码示例。

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的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 AWS 服务结合来完成特定任务的代码示例。

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 SDK 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。

代码示例

- [Amazon EC2 使用的基本示例 AWS SDKs](#)
 - [你好 Amazon EC2](#)
 - [使用 AWS SDK 学习 Amaz EC2 on 的基础知识](#)
 - [Amazon EC2 使用的操作 AWS SDKs](#)
 - [将 AcceptVpcPeeringConnection 与 CLI 配合使用](#)
 - [AllocateAddress与 AWS SDK 或 CLI 配合使用](#)
 - [将 AllocateHosts 与 CLI 配合使用](#)
 - [将 AssignPrivateIpAddresses 与 CLI 配合使用](#)
 - [AssociateAddress与 AWS SDK 或 CLI 配合使用](#)
 - [将 AssociateDhcpOptions 与 CLI 配合使用](#)
 - [将 AssociateRouteTable 与 CLI 配合使用](#)
 - [将 AttachInternetGateway 与 CLI 配合使用](#)
 - [将 AttachNetworkInterface 与 CLI 配合使用](#)
 - [将 AttachVolume 与 CLI 配合使用](#)
 - [将 AttachVpnGateway 与 CLI 配合使用](#)
 - [将 AuthorizeSecurityGroupEgress 与 CLI 配合使用](#)
 - [AuthorizeSecurityGroupIngress与 AWS SDK 或 CLI 配合使用](#)
 - [将 CancelCapacityReservation 与 CLI 配合使用](#)

- [将 CancelImportTask 与 CLI 配合使用](#)
- [将 CancelSpotFleetRequests 与 CLI 配合使用](#)
- [将 CancelSpotInstanceRequests 与 CLI 配合使用](#)
- [将 ConfirmProductInstance 与 CLI 配合使用](#)
- [将 CopyImage 与 CLI 配合使用](#)
- [将 CopySnapshot 与 CLI 配合使用](#)
- [将 CreateCapacityReservation 与 CLI 配合使用](#)
- [将 CreateCustomerGateway 与 CLI 配合使用](#)
- [将 CreateDhcpOptions 与 CLI 配合使用](#)
- [将 CreateFlowLogs 与 CLI 配合使用](#)
- [将 CreateImage 与 CLI 配合使用](#)
- [将 CreateInstanceExportTask 与 CLI 配合使用](#)
- [将 CreateInternetGateway 与 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](#)
- [将 CreateNetworkAcl 与 CLI 配合使用](#)
- [将 CreateNetworkAclEntry 与 CLI 配合使用](#)
- [将 CreateNetworkInterface 与 CLI 配合使用](#)
- [将 CreatePlacementGroup 与 CLI 配合使用](#)
- [将 CreateRoute 与 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](#)
- [将 CreateSnapshot 与 CLI 配合使用](#)
- [将 CreateSpotDatafeedSubscription 与 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](#)
- [将 CreateVolume 与 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](#)
- [将 CreateVpnConnection 与 CLI 配合使用](#)

- [将 CreateVpnConnectionRoute 与 CLI 配合使用](#)
- [将 CreateVpnGateway 与 CLI 配合使用](#)
- [将 DeleteCustomerGateway 与 CLI 配合使用](#)
- [将 DeleteDhcpOptions 与 CLI 配合使用](#)
- [将 DeleteFlowLogs 与 CLI 配合使用](#)
- [将 DeleteInternetGateway 与 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](#)
- [将 DeleteNetworkAcl 与 CLI 配合使用](#)
- [将 DeleteNetworkAclEntry 与 CLI 配合使用](#)
- [将 DeleteNetworkInterface 与 CLI 配合使用](#)
- [将 DeletePlacementGroup 与 CLI 配合使用](#)
- [将 DeleteRoute 与 CLI 配合使用](#)
- [将 DeleteRouteTable 与 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](#)
- [将 DeleteSpotDatafeedSubscription 与 CLI 配合使用](#)
- [将 DeleteSubnet 与 CLI 配合使用](#)
- [将 DeleteTags 与 CLI 配合使用](#)
- [将 DeleteVolume 与 CLI 配合使用](#)
- [DeleteVpc与 AWS SDK 或 CLI 配合使用](#)
- [DeleteVpcEndpoints与 AWS SDK 或 CLI 配合使用](#)
- [将 DeleteVpnConnection 与 CLI 配合使用](#)
- [将 DeleteVpnConnectionRoute 与 CLI 配合使用](#)
- [将 DeleteVpnGateway 与 CLI 配合使用](#)
- [将 DeregisterImage 与 CLI 配合使用](#)
- [将 DescribeAccountAttributes 与 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](#)
- [将 DescribeBundleTasks 与 CLI 配合使用](#)

- [将 DescribeCapacityReservations 与 CLI 配合使用](#)
- [将 DescribeCustomerGateways 与 CLI 配合使用](#)
- [将 DescribeDhcpOptions 与 CLI 配合使用](#)
- [将 DescribeFlowLogs 与 CLI 配合使用](#)
- [将 DescribeHostReservationOfferings 与 CLI 配合使用](#)
- [将 DescribeHosts 与 CLI 配合使用](#)
- [DescribeIamInstanceProfileAssociations 与 AWS SDK 或 CLI 配合使用](#)
- [将 DescribeIdFormat 与 CLI 配合使用](#)
- [将 DescribeIdentityIdFormat 与 CLI 配合使用](#)
- [将 DescribeImageAttribute 与 CLI 配合使用](#)
- [DescribeImages 与 AWS SDK 或 CLI 配合使用](#)
- [将 DescribeImportImageTasks 与 CLI 配合使用](#)
- [将 DescribeImportSnapshotTasks 与 CLI 配合使用](#)
- [将 DescribeInstanceAttribute 与 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](#)
- [将 DescribeInternetGateways 与 CLI 配合使用](#)
- [DescribeKeyPairs 与 AWS SDK 或 CLI 配合使用](#)
- [将 DescribeNetworkAcls 与 CLI 配合使用](#)
- [将 DescribeNetworkInterfaceAttribute 与 CLI 配合使用](#)
- [将 DescribeNetworkInterfaces 与 CLI 配合使用](#)
- [将 DescribePlacementGroups 与 CLI 配合使用](#)
- [将 DescribePrefixLists 与 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](#)
- [将 DescribeScheduledInstanceAvailability 与 CLI 配合使用](#)
- [将 DescribeScheduledInstances 与 CLI 配合使用](#)
- [DescribeSecurityGroups 与 AWS SDK 或 CLI 配合使用](#)
- [将 DescribeSnapshotAttribute 与 CLI 配合使用](#)

- [DescribeSnapshots](#)与 AWS SDK 或 CLI 配合使用
- [将 DescribeSpotDatafeedSubscription 与 CLI 配合使用](#)
- [将 DescribeSpotFleetInstances 与 CLI 配合使用](#)
- [将 DescribeSpotFleetRequestHistory 与 CLI 配合使用](#)
- [将 DescribeSpotFleetRequests 与 CLI 配合使用](#)
- [将 DescribeSpotInstanceRequests 与 CLI 配合使用](#)
- [将 DescribeSpotPriceHistory 与 CLI 配合使用](#)
- [DescribeSubnets](#)与 AWS SDK 或 CLI 配合使用
- [将 DescribeTags 与 CLI 配合使用](#)
- [将 DescribeVolumeAttribute 与 CLI 配合使用](#)
- [将 DescribeVolumeStatus 与 CLI 配合使用](#)
- [将 DescribeVolumes 与 CLI 配合使用](#)
- [将 DescribeVpcAttribute 与 CLI 配合使用](#)
- [将 DescribeVpcClassicLink 与 CLI 配合使用](#)
- [将 DescribeVpcClassicLinkDnsSupport 与 CLI 配合使用](#)
- [将 DescribeVpcEndpointServices 与 CLI 配合使用](#)
- [将 DescribeVpcEndpoints 与 CLI 配合使用](#)
- [DescribeVpcs](#)与 AWS SDK 或 CLI 配合使用
- [将 DescribeVpnConnections 与 CLI 配合使用](#)
- [将 DescribeVpnGateways 与 CLI 配合使用](#)
- [将 DetachInternetGateway 与 CLI 配合使用](#)
- [将 DetachNetworkInterface 与 CLI 配合使用](#)
- [将 DetachVolume 与 CLI 配合使用](#)
- [将 DetachVpnGateway 与 CLI 配合使用](#)
- [将 DisableVgwRoutePropagation 与 CLI 配合使用](#)
- [将 DisableVpcClassicLink 与 CLI 配合使用](#)
- [将 DisableVpcClassicLinkDnsSupport 与 CLI 配合使用](#)
- [DisassociateAddress](#)与 AWS SDK 或 CLI 配合使用
- [将 DisassociateRouteTable 与 CLI 配合使用](#)
- [将 EnableVgwRoutePropagation 与 CLI 配合使用](#)

- [将 EnableVolumelo 与 CLI 配合使用](#)
- [将 EnableVpcClassicLink 与 CLI 配合使用](#)
- [将 EnableVpcClassicLinkDnsSupport 与 CLI 配合使用](#)
- [将 GetConsoleOutput 与 CLI 配合使用](#)
- [将 GetHostReservationPurchasePreview 与 CLI 配合使用](#)
- [GetPasswordData与 AWS SDK 或 CLI 配合使用](#)
- [将 ImportImage 与 CLI 配合使用](#)
- [将 ImportKeyPair 与 CLI 配合使用](#)
- [将 ImportSnapshot 与 CLI 配合使用](#)
- [将 ModifyCapacityReservation 与 CLI 配合使用](#)
- [将 ModifyHosts 与 CLI 配合使用](#)
- [将 ModifyIdFormat 与 CLI 配合使用](#)
- [将 ModifyImageAttribute 与 CLI 配合使用](#)
- [将 ModifyInstanceAttribute 与 CLI 配合使用](#)
- [将 ModifyInstanceCreditSpecification 与 CLI 配合使用](#)
- [将 ModifyNetworkInterfaceAttribute 与 CLI 配合使用](#)
- [将 ModifyReservedInstances 与 CLI 配合使用](#)
- [将 ModifySnapshotAttribute 与 CLI 配合使用](#)
- [将 ModifySpotFleetRequest 与 CLI 配合使用](#)
- [将 ModifySubnetAttribute 与 CLI 配合使用](#)
- [将 ModifyVolumeAttribute 与 CLI 配合使用](#)
- [将 ModifyVpcAttribute 与 CLI 配合使用](#)
- [MonitorInstances与 AWS SDK 或 CLI 配合使用](#)
- [将 MoveAddressToVpc 与 CLI 配合使用](#)
- [将 PurchaseHostReservation 与 CLI 配合使用](#)
- [将 PurchaseScheduledInstances 与 CLI 配合使用](#)
- [RebootInstances与 AWS SDK 或 CLI 配合使用](#)
- [将 RegisterImage 与 CLI 配合使用](#)
- [将 RejectVpcPeeringConnection 与 CLI 配合使用](#)
- [ReleaseAddress与 AWS SDK 或 CLI 配合使用](#)

- [将 ReleaseHosts 与 CLI 配合使用](#)
- [ReplaceIamInstanceProfileAssociation 与 AWS SDK 或 CLI 配合使用](#)
- [将 ReplaceNetworkAclAssociation 与 CLI 配合使用](#)
- [将 ReplaceNetworkAclEntry 与 CLI 配合使用](#)
- [将 ReplaceRoute 与 CLI 配合使用](#)
- [将 ReplaceRouteTableAssociation 与 CLI 配合使用](#)
- [将 ReportInstanceState 与 CLI 配合使用](#)
- [将 RequestSpotFleet 与 CLI 配合使用](#)
- [将 RequestSpotInstances 与 CLI 配合使用](#)
- [将 ResetImageAttribute 与 CLI 配合使用](#)
- [将 ResetInstanceAttribute 与 CLI 配合使用](#)
- [将 ResetNetworkInterfaceAttribute 与 CLI 配合使用](#)
- [将 ResetSnapshotAttribute 与 CLI 配合使用](#)
- [将 RevokeSecurityGroupEgress 与 CLI 配合使用](#)
- [将 RevokeSecurityGroupIngress 与 CLI 配合使用](#)
- [RunInstances 与 AWS SDK 或 CLI 配合使用](#)
- [将 RunScheduledInstances 与 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](#)
- [将 UnassignPrivateIpAddresses 与 CLI 配合使用](#)
- [UnmonitorInstances 与 AWS SDK 或 CLI 配合使用](#)
- [将 UpdateSecurityGroupRuleDescriptionsIngress 与 CLI 配合使用](#)
- [Amazon EC2 使用的场景 AWS SDKs](#)
 - [使用 AWS SDK 构建和管理弹性服务](#)
 - [使用 CLI 创建具有私有子网和 NAT 网关的 VPC。](#)
 - [开始通过 CLI 使用 Amazon VPC](#)
 - [使用 CLI 的中转网关入门](#)
 - [开始通过 CLI 使用 Amazon VPC IPAM](#)

Amazon EC2 使用的基本示例 AWS SDKs

以下代码示例展示了如何使用 Amazon Elastic Compute Cloud 的基础知识 AWS SDKs。

示例

- [你好 Amazon EC2](#)
- [使用 AWS SDK 学习 Amaz EC2 on 的基础知识](#)
- [Amazon EC2 使用的操作 AWS SDKs](#)
 - [将 AcceptVpcPeeringConnection 与 CLI 配合使用](#)
 - [AllocateAddress与 AWS SDK 或 CLI 配合使用](#)
 - [将 AllocateHosts 与 CLI 配合使用](#)
 - [将 AssignPrivateIpAddresses 与 CLI 配合使用](#)
 - [AssociateAddress与 AWS SDK 或 CLI 配合使用](#)
 - [将 AssociateDhcpOptions 与 CLI 配合使用](#)
 - [将 AssociateRouteTable 与 CLI 配合使用](#)
 - [将 AttachInternetGateway 与 CLI 配合使用](#)
 - [将 AttachNetworkInterface 与 CLI 配合使用](#)
 - [将 AttachVolume 与 CLI 配合使用](#)
 - [将 AttachVpnGateway 与 CLI 配合使用](#)
 - [将 AuthorizeSecurityGroupEgress 与 CLI 配合使用](#)
 - [AuthorizeSecurityGroupIngress与 AWS SDK 或 CLI 配合使用](#)
 - [将 CancelCapacityReservation 与 CLI 配合使用](#)
 - [将 CancellImportTask 与 CLI 配合使用](#)
 - [将 CancelSpotFleetRequests 与 CLI 配合使用](#)
 - [将 CancelSpotInstanceRequests 与 CLI 配合使用](#)
 - [将 ConfirmProductInstance 与 CLI 配合使用](#)
 - [将 CopyImage 与 CLI 配合使用](#)
 - [将 CopySnapshot 与 CLI 配合使用](#)
 - [将 CreateCapacityReservation 与 CLI 配合使用](#)
 - [将 CreateCustomerGateway 与 CLI 配合使用](#)
 - [将 CreateDhcpOptions 与 CLI 配合使用](#)

- [将 CreateFlowLogs 与 CLI 配合使用](#)
- [将 CreateImage 与 CLI 配合使用](#)
- [将 CreateInstanceExportTask 与 CLI 配合使用](#)
- [将 CreateInternetGateway 与 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](#)
- [将 CreateNetworkAcl 与 CLI 配合使用](#)
- [将 CreateNetworkAclEntry 与 CLI 配合使用](#)
- [将 CreateNetworkInterface 与 CLI 配合使用](#)
- [将 CreatePlacementGroup 与 CLI 配合使用](#)
- [将 CreateRoute 与 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](#)
- [将 CreateSnapshot 与 CLI 配合使用](#)
- [将 CreateSpotDatafeedSubscription 与 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](#)
- [将 CreateVolume 与 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](#)
- [将 CreateVpnConnection 与 CLI 配合使用](#)
- [将 CreateVpnConnectionRoute 与 CLI 配合使用](#)
- [将 CreateVpnGateway 与 CLI 配合使用](#)
- [将 DeleteCustomerGateway 与 CLI 配合使用](#)
- [将 DeleteDhcpOptions 与 CLI 配合使用](#)
- [将 DeleteFlowLogs 与 CLI 配合使用](#)
- [将 DeleteInternetGateway 与 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](#)
- [将 DeleteNetworkAcl 与 CLI 配合使用](#)

- [将 DeleteNetworkAclEntry 与 CLI 配合使用](#)
- [将 DeleteNetworkInterface 与 CLI 配合使用](#)
- [将 DeletePlacementGroup 与 CLI 配合使用](#)
- [将 DeleteRoute 与 CLI 配合使用](#)
- [将 DeleteRouteTable 与 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](#)
- [将 DeleteSpotDatafeedSubscription 与 CLI 配合使用](#)
- [将 DeleteSubnet 与 CLI 配合使用](#)
- [将 DeleteTags 与 CLI 配合使用](#)
- [将 DeleteVolume 与 CLI 配合使用](#)
- [DeleteVpc与 AWS SDK 或 CLI 配合使用](#)
- [DeleteVpcEndpoints与 AWS SDK 或 CLI 配合使用](#)
- [将 DeleteVpnConnection 与 CLI 配合使用](#)
- [将 DeleteVpnConnectionRoute 与 CLI 配合使用](#)
- [将 DeleteVpnGateway 与 CLI 配合使用](#)
- [将 DeregisterImage 与 CLI 配合使用](#)
- [将 DescribeAccountAttributes 与 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](#)
- [将 DescribeBundleTasks 与 CLI 配合使用](#)
- [将 DescribeCapacityReservations 与 CLI 配合使用](#)
- [将 DescribeCustomerGateways 与 CLI 配合使用](#)
- [将 DescribeDhcpOptions 与 CLI 配合使用](#)
- [将 DescribeFlowLogs 与 CLI 配合使用](#)
- [将 DescribeHostReservationOfferings 与 CLI 配合使用](#)
- [将 DescribeHosts 与 CLI 配合使用](#)
- [DescribelamInstanceProfileAssociations与 AWS SDK 或 CLI 配合使用](#)
- [将 DescribeIdFormat 与 CLI 配合使用](#)
- [将 DescribeIdentityIdFormat 与 CLI 配合使用](#)

- [将 DescribeImageAttribute 与 CLI 配合使用](#)
- [DescribeImages与 AWS SDK 或 CLI 配合使用](#)
- [将 DescribeImportImageTasks 与 CLI 配合使用](#)
- [将 DescribeImportSnapshotTasks 与 CLI 配合使用](#)
- [将 DescribeInstanceAttribute 与 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](#)
- [将 DescribeInternetGateways 与 CLI 配合使用](#)
- [DescribeKeyPairs与 AWS SDK 或 CLI 配合使用](#)
- [将 DescribeNetworkAcls 与 CLI 配合使用](#)
- [将 DescribeNetworkInterfaceAttribute 与 CLI 配合使用](#)
- [将 DescribeNetworkInterfaces 与 CLI 配合使用](#)
- [将 DescribePlacementGroups 与 CLI 配合使用](#)
- [将 DescribePrefixLists 与 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](#)
- [将 DescribeScheduledInstanceAvailability 与 CLI 配合使用](#)
- [将 DescribeScheduledInstances 与 CLI 配合使用](#)
- [DescribeSecurityGroups与 AWS SDK 或 CLI 配合使用](#)
- [将 DescribeSnapshotAttribute 与 CLI 配合使用](#)
- [DescribeSnapshots与 AWS SDK 或 CLI 配合使用](#)
- [将 DescribeSpotDatafeedSubscription 与 CLI 配合使用](#)
- [将 DescribeSpotFleetInstances 与 CLI 配合使用](#)
- [将 DescribeSpotFleetRequestHistory 与 CLI 配合使用](#)
- [将 DescribeSpotFleetRequests 与 CLI 配合使用](#)
- [将 DescribeSpotInstanceRequests 与 CLI 配合使用](#)
- [将 DescribeSpotPriceHistory 与 CLI 配合使用](#)
- [DescribeSubnets与 AWS SDK 或 CLI 配合使用](#)
- [将 DescribeTags 与 CLI 配合使用](#)

- [将 DescribeVolumeAttribute 与 CLI 配合使用](#)
- [将 DescribeVolumeStatus 与 CLI 配合使用](#)
- [将 DescribeVolumes 与 CLI 配合使用](#)
- [将 DescribeVpcAttribute 与 CLI 配合使用](#)
- [将 DescribeVpcClassicLink 与 CLI 配合使用](#)
- [将 DescribeVpcClassicLinkDnsSupport 与 CLI 配合使用](#)
- [将 DescribeVpcEndpointServices 与 CLI 配合使用](#)
- [将 DescribeVpcEndpoints 与 CLI 配合使用](#)
- [DescribeVpcs与 AWS SDK 或 CLI 配合使用](#)
- [将 DescribeVpnConnections 与 CLI 配合使用](#)
- [将 DescribeVpnGateways 与 CLI 配合使用](#)
- [将 DetachInternetGateway 与 CLI 配合使用](#)
- [将 DetachNetworkInterface 与 CLI 配合使用](#)
- [将 DetachVolume 与 CLI 配合使用](#)
- [将 DetachVpnGateway 与 CLI 配合使用](#)
- [将 DisableVgwRoutePropagation 与 CLI 配合使用](#)
- [将 DisableVpcClassicLink 与 CLI 配合使用](#)
- [将 DisableVpcClassicLinkDnsSupport 与 CLI 配合使用](#)
- [DisassociateAddress与 AWS SDK 或 CLI 配合使用](#)
- [将 DisassociateRouteTable 与 CLI 配合使用](#)
- [将 EnableVgwRoutePropagation 与 CLI 配合使用](#)
- [将 EnableVolumelo 与 CLI 配合使用](#)
- [将 EnableVpcClassicLink 与 CLI 配合使用](#)
- [将 EnableVpcClassicLinkDnsSupport 与 CLI 配合使用](#)
- [将 GetConsoleOutput 与 CLI 配合使用](#)
- [将 GetHostReservationPurchasePreview 与 CLI 配合使用](#)
- [GetPasswordData与 AWS SDK 或 CLI 配合使用](#)
- [将 ImportImage 与 CLI 配合使用](#)
- [将 ImportKeyPair 与 CLI 配合使用](#)
- [将 ImportSnapshot 与 CLI 配合使用](#)

- [将 ModifyCapacityReservation 与 CLI 配合使用](#)
- [将 ModifyHosts 与 CLI 配合使用](#)
- [将 ModifyIdFormat 与 CLI 配合使用](#)
- [将 ModifyImageAttribute 与 CLI 配合使用](#)
- [将 ModifyInstanceAttribute 与 CLI 配合使用](#)
- [将 ModifyInstanceCreditSpecification 与 CLI 配合使用](#)
- [将 ModifyNetworkInterfaceAttribute 与 CLI 配合使用](#)
- [将 ModifyReservedInstances 与 CLI 配合使用](#)
- [将 ModifySnapshotAttribute 与 CLI 配合使用](#)
- [将 ModifySpotFleetRequest 与 CLI 配合使用](#)
- [将 ModifySubnetAttribute 与 CLI 配合使用](#)
- [将 ModifyVolumeAttribute 与 CLI 配合使用](#)
- [将 ModifyVpcAttribute 与 CLI 配合使用](#)
- [MonitorInstances 与 AWS SDK 或 CLI 配合使用](#)
- [将 MoveAddressToVpc 与 CLI 配合使用](#)
- [将 PurchaseHostReservation 与 CLI 配合使用](#)
- [将 PurchaseScheduledInstances 与 CLI 配合使用](#)
- [RebootInstances 与 AWS SDK 或 CLI 配合使用](#)
- [将 RegisterImage 与 CLI 配合使用](#)
- [将 RejectVpcPeeringConnection 与 CLI 配合使用](#)
- [ReleaseAddress 与 AWS SDK 或 CLI 配合使用](#)
- [将 ReleaseHosts 与 CLI 配合使用](#)
- [ReplaceIamInstanceProfileAssociation 与 AWS SDK 或 CLI 配合使用](#)
- [将 ReplaceNetworkAclAssociation 与 CLI 配合使用](#)
- [将 ReplaceNetworkAclEntry 与 CLI 配合使用](#)
- [将 ReplaceRoute 与 CLI 配合使用](#)
- [将 ReplaceRouteTableAssociation 与 CLI 配合使用](#)
- [将 ReportInstanceStatus 与 CLI 配合使用](#)
- [将 RequestSpotFleet 与 CLI 配合使用](#)
- [将 RequestSpotInstances 与 CLI 配合使用](#)

- [将 ResetImageAttribute 与 CLI 配合使用](#)
- [将 ResetInstanceAttribute 与 CLI 配合使用](#)
- [将 ResetNetworkInterfaceAttribute 与 CLI 配合使用](#)
- [将 ResetSnapshotAttribute 与 CLI 配合使用](#)
- [将 RevokeSecurityGroupEgress 与 CLI 配合使用](#)
- [将 RevokeSecurityGroupIngress 与 CLI 配合使用](#)
- [RunInstances 与 AWS SDK 或 CLI 配合使用](#)
- [将 RunScheduledInstances 与 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](#)
- [将 UnassignPrivateIpAddresses 与 CLI 配合使用](#)
- [UnmonitorInstances 与 AWS SDK 或 CLI 配合使用](#)
- [将 UpdateSecurityGroupRuleDescriptionsIngress 与 CLI 配合使用](#)

你好 Amazon EC2

以下代码示例展示了如何开始使用 Amazon EC2。

.NET

SDK for .NET (v4)

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。

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

```

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

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

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

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

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

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

```

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

- 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](#) 中查找完整示例，了解如何进行设置和运行。

CMakeLists.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) {
    (void)argc;
    (void)argv;

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

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

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

```

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

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

                Aws::String typeString =

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

                Aws::String monitorString =

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

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

        if (!outcome.GetResult().GetNextToken().empty()) {
            request.SetNextToken(outcome.GetResult().GetNextToken());
        } else {

```

```

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

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

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```

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

```

```
DescribeSecurityGroupsRequest request =
DescribeSecurityGroupsRequest.builder()
    .groupNames(groupName)
    .build();

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

- 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](#) 中查找完整示例，了解如何进行设置和运行。

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

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

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

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

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

- 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](#) 中查找完整示例，了解如何进行设置和运行。

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

    Ec2Client.fromEnvironment { 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 Kotlin 的 AWS SDK API 参考。

Python

适用于 Python 的 SDK (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。

```
def hello_ec2(ec2_client):
```

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

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

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

- 有关 API 的详细信息，请参阅适用[DescribeSecurityGroups](#)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](#) 中查找完整示例，了解如何进行设置和运行。


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

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

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

    instances = fetch_instances

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

  private

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

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

  instances
```

```

end

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

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

```

- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考 [DescribeSecurityGroups](#) 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](#) 中查找完整示例，了解如何进行设置和运行。

```

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

```

```

        .send()
        .await;

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

```

- 有关 API 的详细信息，请参阅适用[DescribeSecurityGroups](#)于 Rust 的AWS SDK API 参考。

Swift

适用于 Swift 的 SDK

Note

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

Package.swift 文件。

```

// swift-tools-version: 5.9
//

```

```
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "hello-ec2",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/aws-labs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
        // or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "hello-ec2",
            dependencies: [
                .product(name: "AWSEC2", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")
    ]
)
```

entry.swift 文件。

```
// An example that shows how to use the AWS SDK for Swift to perform a simple
```

```
// operation using Amazon Elastic Compute Cloud (EC2).
//

import ArgumentParser
import Foundation

import AWSEC2

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"

    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
        completion: .list([
            "critical",
            "debug",
            "error",
            "info",
            "notice",
            "trace",
            "warning"
        ])
    )
    var logLevel: String = "error"

    static var configuration = CommandConfiguration(
        commandName: "hello-ec2",
        abstract: ""
        Demonstrates a simple operation using Amazon EC2.
        "",
        discussion: ""
        An example showing how to make a call to Amazon EC2 using the AWS SDK for
        Swift.
        ""
    )

    /// Return an array of strings giving the names of every security group
    /// the user is a member of.
    ///
    /// - Parameter ec2Client: The `EC2Client` to use when calling
    ///   `describeSecurityGroupsPaginated()`.
    ///
    /// - Returns: An array of strings giving the names of every security
```

```
/// group the user is a member of.
func getSecurityGroupNames(ec2Client: EC2Client) async -> [String] {
    let pages = ec2Client.describeSecurityGroupsPaginated(
        input: DescribeSecurityGroupsInput()
    )

    var groupNames: [String] = []

    do {
        for try await page in pages {
            guard let groups = page.securityGroups else {
                print("*** Error: No groups returned.")
                continue
            }

            for group in groups {
                groupNames.append(group.groupName ?? "<unknown>")
            }
        } catch {
            print("*** Error: \(error.localizedDescription)")
        }

        return groupNames
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let ec2Config = try await EC2Client.EC2ClientConfiguration(region:
awsRegion)
        let ec2Client = EC2Client(config: ec2Config)

        let groupNames = await getSecurityGroupNames(ec2Client: ec2Client)

        print("Found \(groupNames.count) security group(s):")

        for group in groupNames {
            print("    \(group)")
        }
    }
}

/// The program's asynchronous entry point.
@main
```

```
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- 有关 API 的详细信息，请参阅适用于 Swift 的 AWS SDK API 参考 [DescribeSecurityGroups](#) 中。

有关 AWS SDK 开发者指南和代码示例的完整列表，请参阅 [使用 AWS SDK 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 SDK 学习 Amazon EC2 on 的基础知识

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

SDK for .NET

Note

还有更多相关信息在 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场景。

```

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

    public static string associationId = null!;
    public static string allocationId = null!;
    public static string instanceId = null!;
    public static string keyPairName = null!;
    public static string groupName = null!;
    public static string tempFileName = null!;
    public static string secGroupId = null!;
    public static bool isInteractive = true;

    /// <summary>
    /// Perform the actions defined for the Amazon EC2 Basics scenario.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EC2 and Amazon Simple Systems
        // Management (Amazon SSM) Service.
        using var host =
        Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonEC2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddTransient<EC2Wrapper>()
                    .AddTransient<SsmWrapper>()
            )
            .Build();

        SetUpServices(host);

        var uniqueName = Guid.NewGuid().ToString();
        keyPairName = "mvp-example-key-pair" + uniqueName;
    }
}

```



```

        groupName = "ec2-scenario-group" + uniqueName;
        var groupDescription = "A security group created for the EC2 Basics
scenario.";

        try
        {
            // Start the scenario.
            _uiMethods.DisplayOverview();
            _uiMethods.PressEnter(isInteractive);

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

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

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

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

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

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

                // Passing an empty string to the DescribeKeyPairs method will
return

                // a list of all existing key pairs.
                var keyPairs = await _ec2Wrapper.DescribeKeyPairs("");
                keyPairs.ForEach(kp =>

```

```
        {
            Console.WriteLine(
                $"{kp.KeyName} created at: {kp.CreateTime} Fingerprint:
{kp.KeyFingerprint}");
        });
    }

    _uiMethods.PressEnter(isInteractive);

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

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

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

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

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

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

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

var images = await _ec2Wrapper.DescribeImages(imageIds);

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

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

var selectedImage = images[choice - 1];

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

i = 1;
instanceTypes.ForEach(instanceType =>
{
    Console.WriteLine($"{i++}\t{instanceType.InstanceType}");
});
if (isInteractive)
{
    do
```

```

        {
            Console.WriteLine("Please select an instance type: ");
            var selImage = Console.ReadLine();
            validNumber = int.TryParse(selImage, out choice);
        } while (!validNumber);
    }

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

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

    _uiMethods.PressEnter(isInteractive);

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

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

    _uiMethods.PressEnter(isInteractive);

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

    await _ec2Wrapper.StopInstances(instanceId);

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

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

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

    Console.WriteLine("\nNotice the change in the SSH information:");
    Console.WriteLine(

```

```
        $"\\tssh -i {tempFileName} ec2-user@{instance.PublicIpAddress}");

        _uiMethods.PressEnter(isInteractive);

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

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

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

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

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

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

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

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

        await _ec2Wrapper.StopInstances(instanceId);
```

```

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

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

        instance = await _ec2Wrapper.DescribeInstance(instanceId);
        _uiMethods.DisplayTitle("New Instance Information");
        _ec2Wrapper.DisplayInstanceInformation(instance);
        Console.WriteLine("Note that the IP address did not change this
time.");
        _uiMethods.PressEnter(isInteractive);

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

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

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

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

/// <summary>

```

```

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

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

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

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

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

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

```

定义一个封装 EC2 动作的类。

```

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

```

```
/// <summary>
/// Constructor for the EC2Wrapper class.
/// </summary>
/// <param name="amazonScheduler">The injected EC2 client.</param>
/// <param name="logger">The injected logger.</param>
public EC2Wrapper(IAmazonEC2 amazonService, ILogger<EC2Wrapper> logger)
{
    _amazonEC2 = amazonService;
    _logger = logger;
}

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

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

        throw;
    }
    catch (Exception ex)
    {
    }
}
```



```

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

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

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

            throw;
        }
    }

```

```

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

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

```

```

        }

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

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

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

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

        var response = await _amazonEC2.CreateKeyPairAsync(request);

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

```

```

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

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

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

/// <summary>
/// Save KeyPair information to a temporary file.
/// </summary>
/// <param name="keyPair">The name of the key pair.</param>
/// <returns>The full path to the temporary file.</returns>
public string SaveKeyPair(KeyPair keyPair)
{

```

```

        var tempPath = Path.GetTempPath();
        var tempFileName = $"{tempPath}\\{Path.GetRandomFileName()}";
        var pemFileName = Path.ChangeExtension(tempFileName, "pem");

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

        return pemFileName;
    }

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

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

                Thread.Sleep(5000);
                retries--;
            }
            _logger.LogError($"Unable to find newly created group {groupName}.");
            throw new DoesNotExistException("security group not found");
        }
    }

```

```

        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "ResourceAlreadyExists")
            {
                _logger.LogError(
                    $"A security group with the name {groupName} already exists.
{ec2Exception.Message}");
            }
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(
                $"An error occurred while creating the security group.:
{ex.Message}");
            throw;
        }
    }
}

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

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

```

```
    }  
}  
  
/// <summary>  
/// Delete an Amazon EC2 key pair.  
/// </summary>  
/// <param name="keyPairName">The name of the key pair to delete.</param>  
/// <returns>A Boolean value indicating the success of the action.</returns>  
public async Task<bool> DeleteKeyPair(string keyPairName)  
{  
    try  
    {  
        await _amazonEC2.DeleteKeyPairAsync(new  
DeleteKeyPairRequest(keyPairName)).ConfigureAwait(false);  
        return true;  
    }  
    catch (AmazonEC2Exception ec2Exception)  
    {  
        if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")  
        {  
            _logger.LogError($"KeyPair {keyPairName} does not exist and  
cannot be deleted. Please verify the key pair name and try again.");  
        }  
  
        return false;  
    }  
    catch (Exception ex)  
    {  
        Console.WriteLine($"Couldn't delete the key pair because:  
{ex.Message}");  
        return false;  
    }  
}  
  
/// <summary>  
/// Delete the temporary file where the key pair information was saved.  
/// </summary>  
/// <param name="tempFileName">The path to the temporary file.</param>  
public void DeleteTempFile(string tempFileName)  
{  
    if (File.Exists(tempFileName))  
    {  
        File.Delete(tempFileName);  
    }  
}
```

```
}

/// <summary>
/// Delete an Amazon EC2 security group.
/// </summary>
/// <param name="groupName">The name of the group to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteSecurityGroup(string groupId)
{
    try
    {
        var response =
            await _amazonEC2.DeleteSecurityGroupAsync(
                new DeleteSecurityGroupRequest { GroupId = groupId });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
        {
            _logger.LogError(
                $"Security Group {groupId} does not exist and cannot be
deleted. Please verify the ID and try again.");
        }

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

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



```

    };

    var response = await _amazonEC2.DeleteVpcAsync(request);

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

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

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

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

/// <summary>
/// Get information about an Amazon EC2 instance.
/// </summary>
/// <param name="instanceId">The instance Id of the EC2 instance.</param>
/// <returns>An EC2 instance.</returns>
public async Task<Instance> DescribeInstance(string instanceId)

```

```

    {
        var response = await _amazonEC2.DescribeInstancesAsync(
            new DescribeInstancesRequest { InstanceIds = new List<string>
{ instanceId } });
        return response.Reservations[0].Instances[0];
    }

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

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

            Console.WriteLine($"\\nShowing instances with state {state}");

```

```

        var paginator = _amazonEC2.Paginators.DescribeInstances(request);

        await foreach (var response in paginator.Responses)
        {
            foreach (var reservation in response.Reservations)
            {
                foreach (var instance in reservation.Instances)
                {
                    Console.WriteLine($"Instance ID: {instance.InstanceId} ");
                    Console.WriteLine($"\\tCurrent State:
{instance.State.Name}");
                }
            }
        }

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

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

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

```

```

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

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

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

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

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

/// <summary>
/// Get information about an Amazon EC2 key pair.
/// </summary>
/// <param name="keyPairName">The name of the key pair.</param>
/// <returns>A list of key pair information.</returns>

```

```

public async Task<List<KeyPairInfo>> DescribeKeyPairs(string keyPairName)
{
    try
    {
        var request = new DescribeKeyPairsRequest();
        if (!string.IsNullOrEmpty(keyPairName))
        {
            request = new DescribeKeyPairsRequest
            {
                KeyNames = new List<string> { keyPairName }
            };
        }

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

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

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

```

```
try
{
    var securityGroups = new List<SecurityGroup>();
    var request = new DescribeSecurityGroupsRequest();

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

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

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

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

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

/// <summary>
/// Display the information returned by the call to
```

```

/// DescribeSecurityGroupsAsync.
/// </summary>
/// <param name="securityGroup">A list of security group information.</param>
public void DisplaySecurityGroupInfoAsync(SecurityGroup securityGroup)
{
    Console.WriteLine($"{securityGroup.GroupName}");
    Console.WriteLine("Ingress permissions:");
    securityGroup.IpPermissions.ForEach(permission =>
    {
        Console.WriteLine($"  \tFromPort: {permission.FromPort}");
        Console.WriteLine($"  \tIpProtocol: {permission.IpProtocol}");

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

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

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

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

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

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

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

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

```

```

    }

    /// <summary>
    /// Disassociate an Elastic IP address from an EC2 instance.
    /// </summary>
    /// <param name="associationId">The association Id.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DisassociateIp(string associationId)
    {
        try
        {
            var response = await _amazonEC2.DisassociateAddressAsync(
                new DisassociateAddressRequest { AssociationId =
associationId });
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidAssociationID.NotFound")
            {
                _logger.LogError(
                    $"AssociationId is invalid, unable to disassociate address.
{ec2Exception.Message}");
            }

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

    /// <summary>
    /// Reboot a specific EC2 instance.
    /// </summary>
    /// <param name="ec2InstanceId">The instance Id of the instance that will be
rebooted.</param>
    /// <returns>Async Task.</returns>
    public async Task<bool> RebootInstances(string ec2InstanceId)

```



```

{
    try
    {
        var request = new RebootInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId },
        };

        await _amazonEC2.RebootInstancesAsync(request);

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

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

/// <summary>
/// Release an Elastic IP address. After the Elastic IP address is released,
/// it can no longer be used.
/// </summary>
/// <param name="allocationId">The allocation Id of the Elastic IP address.</
param>
/// <returns>True if successful.</returns>
public async Task<bool> ReleaseAddress(string allocationId)
{

```

```

        try
        {
            var request = new ReleaseAddressRequest { AllocationId =
allocationId };

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

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

    /// <summary>
    /// Create and run an EC2 instance.
    /// </summary>
    /// <param name="ImageId">The image Id of the image used as a basis for the
    /// EC2 instance.</param>
    /// <param name="instanceType">The instance type of the EC2 instance to
    create.</param>
    /// <param name="keyName">The name of the key pair to associate with the
    /// instance.</param>
    /// <param name="groupId">The Id of the Amazon EC2 security group that will
    be
    /// allowed to interact with the new EC2 instance.</param>
    /// <returns>The instance Id of the new EC2 instance.</returns>
    public async Task<string> RunInstances(string imageId, string instanceType,
string keyName, string groupId)
    {

```

```
try
{
    var request = new RunInstancesRequest
    {
        ImageId = imageId,
        InstanceType = instanceType,
        KeyName = keyName,
        MinCount = 1,
        MaxCount = 1,
        SecurityGroupIds = new List<string> { groupId }
    };
    var response = await _amazonEC2.RunInstancesAsync(request);
    var instanceId = response.Reservation.Instances[0].InstanceId;

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

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

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



```
/// <summary>
/// Start an EC2 instance.
/// </summary>
/// <param name="ec2InstanceId">The instance Id of the Amazon EC2 instance
/// to start.</param>
/// <returns>Async task.</returns>
```

```

public async Task StartInstances(string ec2InstanceId)
{
    try
    {
        var request = new StartInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId },
        };

        await _amazonEC2.StartInstancesAsync(request);

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

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

/// <summary>
/// Stop an EC2 instance.
/// </summary>
/// <param name="ec2InstanceId">The instance Id of the EC2 instance to
/// stop.</param>
/// <returns>Async task.</returns>
public async Task StopInstances(string ec2InstanceId)
{
    try
    {
        var request = new StopInstancesRequest

```

```

        {
            InstanceIds = new List<string> { ec2InstanceId },
        };

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

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

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

/// <summary>
/// Terminate an EC2 instance.
/// </summary>
/// <param name="ec2InstanceId">The instance Id of the EC2 instance
/// to terminate.</param>
/// <returns>Async task.</returns>
public async Task<List<InstanceStateChange>> TerminateInstances(string
ec2InstanceId)
{
    try
    {
        var request = new TerminateInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId }
        };
    }

```

```

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

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

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

/// <summary>
/// Wait until an EC2 instance is in a specified state.
/// </summary>
/// <param name="instanceId">The instance Id.</param>
/// <param name="stateName">The state to wait for.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
{
    var request = new DescribeInstancesRequest
    {
        InstanceIds = new List<string> { instanceId }
    };

```

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

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

return hasState;
}
}
```

- 有关 API 详细信息，请参阅《AWS SDK for .NET API Reference》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。

```
#####
# 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
}

```

```

fi

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

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

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

return $result
}

#####
# function ssm_get_parameters_by_path
#
# This function retrieves one or more parameters from the AWS Systems Manager
# Parameter Store
# by specifying a parameter path.
#
# Parameters:
#     -p parameter_path - The path of the parameter(s) to retrieve.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function ssm_get_parameters_by_path() {

```

```

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

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

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

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

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

```

```

    echo "$response"

    return 0
}

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

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

    local instance_id image_id instance_type key_name vpc_id public_ip state
    instance_id=$(echo "${instance_details}" | awk '{print $1}')
    image_id=$(echo "${instance_details}" | awk '{print $2}')
    instance_type=$(echo "${instance_details}" | awk '{print $3}')
    key_name=$(echo "${instance_details}" | awk '{print $4}')
    vpc_id=$(echo "${instance_details}" | awk '{print $5}')
    public_ip=$(echo "${instance_details}" | awk '{print $6}')
    state=$(echo "${instance_details}" | awk '{print $7}')

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

```

```

    return 0
}

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

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

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

    # Display the public IP address and connection command
    echo "To connect, run the following command:"
    echo "    ssh -i ${key_file_name} ec2-user@${public_ip}"

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

```

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

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

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

    return 0
}

#####
# function yes_no_input
#
# This function requests a yes/no answer from the user, following to a prompt.
#
# Parameters:
#   $1 - The prompt.
#
# Returns:
#   0 - If yes.
#   1 - If no.
```

```
#####
function yes_no_input() {
    if [ -z "$1" ]; then
        echo "Internal error yes_no_input"
        return 1
    fi

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

    echo

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

#####
# function integer_input
#
# This function prompts the user to enter an integer within a specified range
# and validates the input.
#
# Parameters:
#     $1 - The prompt message to display to the user.
#     $2 - The minimum value of the accepted range.
#     $3 - The maximum value of the accepted range.
#
# Returns:
```

```

#       The valid integer input from the user.
#       If the input is invalid or out of range, the function will continue
#       prompting the user until a valid input is provided.
#####
function integer_input() {
    local prompt="$1"
    local min_value="$2"
    local max_value="$3"
    local input=""

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

        if ! get_input; then
            return 1
        fi

        input="$get_input_result"

        # Check if the input is a valid integer
        if [[ "$input" =~ ^-?[0-9]+$ ]]; then
            # Check if the input is within the specified range
            if ((input >= min_value && input <= max_value)); then
                return 0
            else
                echo "Error: Input, $input, must be between $min_value and $max_value."
            fi
        else
            echo "Error: Invalid input- $input. Please enter an integer."
        fi
    done
}
#####
# function new_line_and_tab_to_list
#
# This function takes a string input containing newlines and tabs, and
# converts it into a list (array) of elements.
#
# Parameters:
#     $1 - The input string containing newlines and tabs.
#
# Returns:
#     The resulting list (array) is stored in the global variable

```



```
# 'list_result'.
#####
function new_line_and_tab_to_list() {
    local input=$1
    export list_result

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

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

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

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

```

```

export OPTIND=1

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

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

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

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

return 0
}

#####
# function ec2_describe_key_pairs
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
# key pairs.
#
# Parameters:
#     -h - Display help.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_describe_key_pairs() {

```

```

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

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

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

local response

response=$(aws ec2 describe-key-pairs \
    --query 'KeyPairs[*].[KeyName, KeyFingerprint]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports describe-key-pairs operation failed.$response"
    return 1
}

echo "$response"

return 0
}

#####
# function ec2_create_security_group

```

```

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

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

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

```

```

done
export OPTIND=1

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

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

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

echo "$response"
return 0
}

#####
# function ec2_describe_security_groups
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
# security groups.
#
# Parameters:
#     -g security_group_id - The ID of the security group to describe
#     (optional).
#
# And:
#     0 - If successful.

```

```

#      1 - If it fails.
#####
function ec2_describe_security_groups() {
    local security_group_id response
    local option OPTARG # Required to use getopt command in a function.

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

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

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

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

    local error_code=${?}

```

```

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

    echo "$response"

    return 0
}

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

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

```



```
    echo ""
}

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

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

if [[ -z "$ip_address" ]]; then
    errecho "ERROR: You must provide an IP address or CIDR block with the -i
parameter."
    usage
    return 1
fi

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

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

```

    errecho "ERROR: You must provide a start port with the -f parameter."
    usage
    return 1
fi

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

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

return 0
}

#####
# function ec2_describe_images
#
# This function describes one or more Amazon Elastic Compute Cloud (Amazon EC2)
# images.
#
# Parameters:
#     -i image_ids - A space-separated list of image IDs (optional).
#     -h - Display help.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_describe_images() {
    local image_ids response
    local option OPTARG # Required to use getopt command in a function.

```

```

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

# Retrieve the calling parameters.
while 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
                ;;
        esac
    done
}

```

```

        -h | --help)
            usage
            return 0
            ;;
        *)
            echo "Unknown argument: $1"
            return 1
            ;;
    esac
done

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

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

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

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

```

```

local array_size
array_size=${#items[@]}
for i in $(seq 0 $((array_size - 1))); do
    echo -n '"${items[$i]}"' >>"$tmp_json_file"
    if [[ $i -lt $((array_size - 1)) ]]; then
        echo -n ',' >>"$tmp_json_file"
    fi
done

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

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

local error_code=$?

rm "$tmp_json_file"

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

echo "$response"
return 0
}

#####
# function ec2_run_instances
#
# This function launches one or more Amazon Elastic Compute Cloud (Amazon EC2)
# instances.
#
# Parameters:
#     -i image_id - The ID of the Amazon Machine Image (AMI) to use.
#     -t instance_type - The instance type to use (e.g., t2.micro).
#     -k key_pair_name - The name of the key pair to use.
#     -s security_group_id - The ID of the security group to use.
#     -c count - The number of instances to launch (default: 1).
#     -h - Display help.
#
# Returns:

```

```

#      0 - If successful.
#      1 - If it fails.
#####
function ec2_run_instances() {
    local image_id instance_type key_pair_name security_group_id count response
    local option OPTARG # Required to use getopt command in a function.

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

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

    if [[ -z "$image_id" ]]; then

```

```
    errecho "ERROR: You must provide an Amazon Machine Image (AMI) ID with the -i
parameter."
    usage
    return 1
fi

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

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

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

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

response=$(aws ec2 run-instances \
    --image-id "$image_id" \
    --instance-type "$instance_type" \
    --key-name "$key_pair_name" \
    --security-group-ids "$security_group_id" \
    --count "$count" \
    --query 'Instances[*].[InstanceId]' \
    --output text) || {
    aws_cli_error_log ${?}
    errecho "ERROR: AWS reports run-instances operation failed.$response"
    return 1
}

echo "$response"

return 0
```



```

}

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

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

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

```

```

        return 1
        ;;
    esac
done
export OPTIND=1

local aws_cli_args=()

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

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

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

echo "$response"

return 0
}

#####
# function ec2_stop_instances
#
# This function stops one or more Amazon Elastic Compute Cloud (Amazon EC2)
# instances.
#
# Parameters:
#     -i instance_id - The ID(s) of the instance(s) to stop (comma-separated).

```

```

#       -h - Display help.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
#####
function ec2_stop_instances() {
    local instance_ids
    local option OPTARG # Required to use getopt command in a function.

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

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

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

```

```

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

return 0
}

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

  # bashsupport disable=BP5008
  function usage() {
    echo "function ec2_start_instances"
    echo "Starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
instances."
    echo "  -i instance_id - The ID(s) of the instance(s) to start (comma-
separated)."
    echo "  -h - Display help."
    echo ""
  }

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

```

```

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

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

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

return 0
}

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

```

```
#      1 - If it fails.
#
#####
function ec2_allocate_address() {
    local domain response

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

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

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

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

```

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

echo "$response"
return 0
}

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

  # Function to display usage information
  function usage() {
    echo "function ec2_associate_address"
    echo "Associates an Elastic IP address with an Amazon Elastic Compute Cloud
(Amazon EC2) instance."
    echo "  -a allocation_id - The allocation ID of the Elastic IP address to
associate."
  }

```

```

    echo "  -i instance_id - The ID of the EC2 instance to associate the Elastic
IP address with."
    echo ""
}

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

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

if [[ -z "$instance_id" ]]; then
    errecho "ERROR: You must provide an instance ID with the -i parameter."
    return 1
fi

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

```



```

}

echo "$response"
return 0
}

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

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

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

```

```

        usage
        return 1
    ;;
esac
done
export OPTIND=1

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

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

return 0
}

#####
# function ec2_release_address
#
# This function releases an Elastic IP address from an Amazon Elastic Compute
# Cloud (Amazon EC2) instance.
#
# Parameters:
#     -a allocation_id - The allocation ID of the Elastic IP address to
#     release.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#
#####
function ec2_release_address() {
    local allocation_id response

    # Function to display usage information
    function usage() {

```

```

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

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

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

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

return 0
}

#####
# function ec2_terminate_instances

```

```

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

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

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

    # Check if instance ID is provided

```

```

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

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

return 0
}

#####
# function ec2_delete_security_group
#
# This function deletes an Amazon Elastic Compute Cloud (Amazon EC2) security
# group.
#
# Parameters:
#     -i security_group_id - The ID of the security group to delete.
#
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function ec2_delete_security_group() {
    local security_group_id response
    local option OPTARG # Required to use getopt command in a function.

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

```

```

# Retrieve the calling parameters.
while 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 getopt command in a function.
    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_delete_keypair"
        echo "Deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair."
        echo "  -n key_pair_name - A key pair name."
        echo ""
    }

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

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

在命令提示符中运行场景。

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

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

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Creates an RSA key pair and saves the private key data as a .pem file.
 * 2. Lists key pairs.
```

```

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

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

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

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

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

        String keyName = "TestKeyPair7" ;
        String fileName = "ec2Key.pem";
        String groupName = "TestSecGroup7" ;

```

```
String groupDesc = "Test Group" ;
String vpcId = ec2Actions.describeFirstEC2VpcAsync().join().vpcId();
InetAddress localAddress = InetAddress.getLocalHost();
String myIpAddress = localAddress.getHostAddress();

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

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

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

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

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

logger.info(DASHES);
```

```

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

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

        logger.info(DASHES);
        logger.info("2. List key pairs.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<DescribeKeyPairsResponse> future =
ec2Actions.describeKeysAsync();

```

```

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

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

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

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof Ec2Exception ec2Ex) {
            if (ec2Ex.awsErrorDetails().errorMessage().contains("already
exists")) {

```

```

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

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

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

```

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

        logger.info(DASHES);
        logger.info("6. Get the (Amazon Machine Image) AMI value from the amzn2
image.");
```



```

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

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

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

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

```

```

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

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

```

```

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

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

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

logger.info("You can SSH to the instance using this command:");

```

```
logger.info("ssh -i " + fileName + " ec2-user@" + publicIp);
waitForInputToContinue(scanner);
logger.info(DASHES);

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

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

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

} catch (RuntimeException rt) {
    Throwable cause = rt.getCause();
    if (cause instanceof Ec2Exception ec2Ex) {
        // Handle EC2 exceptions.
        logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
        return;
    }
}
```

```

        } else {
            logger.info("An unexpected error occurred: {}",
cause.getMessage());
            return;
        }
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

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

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

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

        """);

    waitForInputToContinue(scanner);
    String allocationId;
    try {
        CompletableFuture<String> future = ec2Actions.allocateAddressAsync();
        allocationId = future.join();
        logger.info("Successfully allocated address with ID: "
+allocationId);
    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();

```

```

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

    logger.info(DASHES);
    logger.info("13. Describe the instance again. Note that the public IP
address has changed");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<String> future =
ec2Actions.describeEC2InstancesAsync(newInstanceId);
        publicIp = future.join();
        logger.info("EC2 instance public IP: " + publicIp);
        logger.info("You can SSH to the instance using this command:");
    }

```

```

        logger.info("ssh -i " + fileName + " ec2-user@" + publicIp);
    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof Ec2Exception ec2Ex) {
            logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
            return;
        } else {
            logger.info("An unexpected error occurred: {}",
cause.getMessage());
            return;
        }
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("14. Disassociate and release the Elastic IP address.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<DisassociateAddressResponse> future =
ec2Actions.disassociateAddressAsync(associationId);
        future.join();
        logger.info("Address successfully disassociated.");
    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof Ec2Exception ec2Ex) {
            // Handle EC2 exceptions.
            logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
            return;
        } else {
            logger.info("An unexpected error occurred: {}",
cause.getMessage());
            return;
        }
    }
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<ReleaseAddressResponse> future =
ec2Actions.releaseEC2AddressAsync(allocationId);
        future.join(); // Wait for the operation to complete
        logger.info("Elastic IP address successfully released.");
    } catch (RuntimeException rte) {

```

```

        logger.info("An unexpected error occurred: {}", rte.getMessage());
        return;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

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

    logger.info(DASHES);
    logger.info("16. Delete the security group.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<Void> future =
ec2Actions.deleteEC2SecGroupAsync(groupId);
        future.join();
        logger.info("Security group successfully deleted.");
    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof Ec2Exception ec2Ex) {
            logger.info("EC2 error occurred: Message {}, Error Code:{}",
ec2Ex.getMessage(), ec2Ex.awsErrorDetails().errorCode());
            return;
        }
    }

```



```

        } else {
            logger.info("An unexpected error occurred: {}",
cause.getMessage());
            return;
        }
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

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

    logger.info(DASHES);
    logger.info("You successfully completed the Amazon EC2 scenario.");
    logger.info(DASHES);
}

public static boolean filterName(String name) {
    String[] parts = name.split("/");
    String myValue = parts[4];
    return myValue.contains("amzn2");
}

private static void waitForInputToContinue(Scanner scanner) {
    while (true) {

```

```

        logger.info("");
        logger.info("Enter 'c' followed by <ENTER> to continue:");
        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
            logger.info("Continuing with the program...");
            logger.info("");
            break;
        } else {
            // Handle invalid input.
            logger.info("Invalid input. Please try again.");
        }
    }
}
}

```

定义一个封装 EC2 动作的类。

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.AllocateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AllocateAddressResponse;
import software.amazon.awssdk.services.ec2.model.AssociateAddressRequest;
import software.amazon.awssdk.services.ec2.model.AssociateAddressResponse;
import
    software.amazon.awssdk.services.ec2.model.AuthorizeSecurityGroupIngressRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.CreateKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.CreateSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairRequest;
import software.amazon.awssdk.services.ec2.model.DeleteKeyPairResponse;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupRequest;
import software.amazon.awssdk.services.ec2.model.DeleteSecurityGroupResponse;
import software.amazon.awssdk.services.ec2.model.DescribeImagesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstanceTypesRequest;
import software.amazon.awssdk.services.ec2.model.DescribeInstanceTypesResponse;
import software.amazon.awssdk.services.ec2.model.DescribeInstancesRequest;

```

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

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

```

/**
 * Retrieves an asynchronous Amazon Elastic Container Registry (ECR) client.
 *
 * @return the configured ECR asynchronous client.
 */
private static Ec2AsyncClient getAsyncClient() {
    if (ec2AsyncClient == null) {
        /**
         * The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
         version 2,
         and it is designed to provide a high-performance, asynchronous HTTP
         client for interacting with AWS services.
         It uses the Netty framework to handle the underlying network
         communication and the Java NIO API to
         provide a non-blocking, event-driven approach to HTTP requests and
         responses.
         */
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(50) // Adjust as needed.
            .connectionTimeout(Duration.ofSeconds(60)) // Set the connection
timeout.

            .readTimeout(Duration.ofSeconds(60)) // Set the read timeout.
            .writeTimeout(Duration.ofSeconds(60)) // Set the write timeout.
            .build();

        ClientOverrideConfiguration overrideConfig =
        ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2)) // Set the overall API
call timeout.

            .apiCallAttemptTimeout(Duration.ofSeconds(90)) // Set the
individual call attempt timeout.
            .build();

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

/**
 * Deletes a key pair asynchronously.

```

```

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

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

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

        CompletableFuture<DeleteSecurityGroupResponse> response =
getAsyncClient().deleteSecurityGroup(request);
        return response.whenComplete((resp, ex) -> {
            if (ex != null) {

```

```

        throw new RuntimeException("Failed to delete security group with
Id " + groupId, ex);
    } else if (resp == null) {
        throw new RuntimeException("No response received for deleting
security group with Id " + groupId);
    }
}).thenApply(resp -> null);
}

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

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

```

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

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

    return response;
}

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

```

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

        return response;
    }

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

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

```



```

    });
}

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

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

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

    DescribeInstancesPublisher paginator =
getAsyncClient().describeInstancesPaginator(request);
    AtomicReference<String> publicIpAddressRef = new AtomicReference<>();
    return paginator.subscribe(response -> {
        response.reservations().stream()

```

```

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

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

```

```

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

    /**
     * Asynchronously retrieves the instance types available in the current AWS
region.
     * <p>
     * This method uses the AWS SDK's asynchronous API to fetch the available
instance types
     * and then processes the response. It logs the memory information, network
information,
     * and instance type for each instance type returned. Additionally, it
returns a
     * {@link CompletableFuture} that resolves to the instance type string for
the "t2.xlarge"
     * instance type, if it is found in the response. If the "t2.xlarge"
instance type is not
     * found, an empty string is returned.
     * </p>
     *
     * @return a {@link CompletableFuture} that resolves to the instance type
string for the
     * "t2.xlarge" instance type, or an empty string if the instance type is not
found
     */
    public CompletableFuture<String> getInstanceTypesAsync() {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
            .maxResults(10)

```

```

        .build();

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

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

    /**
     * Asynchronously describes an AWS EC2 image with the specified image ID.
     *
     * @param imageId the ID of the image to be described
     * @return a {@link CompletableFuture} that, when completed, contains the ID
of the described image
     * @throws RuntimeException if no images are found with the provided image
ID, or if an error occurs during the AWS API call
     */
    public CompletableFuture<String> describeImageAsync(String imageId) {
        DescribeImagesRequest imagesRequest = DescribeImagesRequest.builder()
            .imageIds(imageId)
            .build();
    }

```

```

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

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

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

```

```

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

        return responseFuture;
    }

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

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

```

```

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

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

    return getAsyncClient().createSecurityGroup(createRequest)
        .thenCompose(createResponse -> {
            String groupId = createResponse.groupId();
            IpRange ipRange = IpRange.builder()

```

```

        .cidrIp(myIpAddress + "/32")
        .build();

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

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

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

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

/**
 * Asynchronously describes the key pairs associated with the current AWS
account.
 *

```



```

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

        return responseFuture;
    }

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

        CompletableFuture<CreateKeyPairResponse> responseFuture =
getAsyncClient().createKeyPair(request);
        responseFuture.whenComplete((response, exception) -> {
            if (response != null) {
                try {
                    BufferedWriter writer = new BufferedWriter(new
FileWriter(fileName));
                    writer.write(response.keyMaterial());
                    writer.close();
                } catch (IOException e) {
                    throw new RuntimeException("Failed to write key material to
file: " + e.getMessage(), e);
                }
            }
        });
    }

```

```

        }
    } else {
        throw new RuntimeException("Failed to create key pair: " +
exception.getMessage(), exception);
    }
});

return responseFuture;
}

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

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

    DescribeVpcsPublisher paginator =
getAsyncClient().describeVpcsPaginator(request);
    AtomicReference<Vpc> vpcRef = new AtomicReference<>();
    return paginator.subscribe(response -> {
        response.vpcs().stream()
            .findFirst()
            .ifPresent(vpcRef::set);
    }).thenApply(v -> {
        Vpc vpc = vpcRef.get();
        if (vpc == null) {
            throw new RuntimeException("Default VPC not found");
        }
        return vpc;
    }).exceptionally(ex -> {
        logger.info("Failed to describe VPCs: " + ex.getMessage());
        throw new RuntimeException("Failed to describe VPCs", ex);
    });
}
}

```

```

/**
 * Stops the EC2 instance with the specified ID asynchronously and waits for
 the instance to stop.
 *
 * @param instanceId the ID of the EC2 instance to stop
 * @return a {@link CompletableFuture} that completes when the instance has
 been stopped, or exceptionally if an error occurs
 */
public CompletableFuture<Void> stopInstanceAsync(String instanceId) {
    StopInstancesRequest stopRequest = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
        .client(getAsyncClient())
        .build();

    CompletableFuture<Void> resultFuture = new CompletableFuture<>();
    logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
    getAsyncClient().stopInstances(stopRequest)
        .thenCompose(response -> {
            if (response.stoppingInstances().isEmpty()) {
                return CompletableFuture.failedFuture(new
RuntimeException("No instances were stopped. Please check the instance ID: " +
instanceId));
            }
            return ec2Waiter.waitUntilInstanceStopped(describeRequest);
        })
        .thenAccept(waiterResponse -> {
            logger.info("Successfully stopped instance " + instanceId);
            resultFuture.complete(null);
        })
        .exceptionally(throwable -> {
            logger.error("Failed to stop instance " + instanceId + ": " +
throwable.getMessage(), throwable);
            resultFuture.completeExceptionally(new RuntimeException("Failed
to stop instance: " + throwable.getMessage(), throwable));
        });
}

```

```

        return null;
    });

    return resultFuture;
}

/**
 * Starts an Amazon EC2 instance asynchronously and waits until it is in the
 * "running" state.
 *
 * @param instanceId the ID of the instance to start
 * @return a {@link CompletableFuture} that completes when the instance has
 * been started and is in the "running" state, or exceptionally if an error occurs
 */
public CompletableFuture<Void> startInstanceAsync(String instanceId) {
    StartInstancesRequest startRequest = StartInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
        .client(getAsyncClient())
        .build();

    DescribeInstancesRequest describeRequest =
DescribeInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    logger.info("Starting instance " + instanceId + " and waiting for it to
run.");
    CompletableFuture<Void> resultFuture = new CompletableFuture<>();
    return getAsyncClient().startInstances(startRequest)
        .thenCompose(response ->
            ec2Waiter.waitUntilInstanceRunning(describeRequest)
        )
        .thenAccept(waiterResponse -> {
            logger.info("Successfully started instance " + instanceId);
            resultFuture.complete(null);
        })
        .exceptionally(throwable -> {
            resultFuture.completeExceptionally(new RuntimeException("Failed
to start instance: " + throwable.getMessage(), throwable));
            return null;
        });
}

```

```
}  
  
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Java 2.x API Reference》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](#) 中查找完整示例，了解如何进行设置和运行。

此文件包含与使用的常用操作的列表 EC2。这些步骤通过场景框架构建，该框架可简化交互式示例的运行。有关完整上下文，请访问 GitHub 存储库。

```
import { tmpdir } from "node:os";
import { writeFile, mkdtemp, rm } from "node:fs/promises";
import { join } from "node:path";
import { get } from "node:http";

import {
  AllocateAddressCommand,
  AssociateAddressCommand,
  AuthorizeSecurityGroupIngressCommand,
  CreateKeyPairCommand,
  CreateSecurityGroupCommand,
  DeleteKeyPairCommand,
  DeleteSecurityGroupCommand,
  DisassociateAddressCommand,
  paginateDescribeImages,
  paginateDescribeInstances,
  paginateDescribeInstanceTypes,
  ReleaseAddressCommand,
  RunInstancesCommand,
  StartInstancesCommand,
  StopInstancesCommand,
  TerminateInstancesCommand,
  waitUntilInstanceStatusOk,
  waitUntilInstanceStopped,
  waitUntilInstanceTerminated,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
```

```

    ScenarioInput,
    ScenarioOutput,
  } from "@aws-doc-sdk-examples/lib/scenario/index.js";

import { paginateGetParametersByPath, SSMClient } from "@aws-sdk/client-ssm";

/**
 * @typedef {{
 *   ec2Client: import('@aws-sdk/client-ec2').EC2Client,
 *   errors: Error[],
 *   keyPairId?: string,
 *   tmpDirectory?: string,
 *   securityGroupId?: string,
 *   ipAddress?: string,
 *   images?: import('@aws-sdk/client-ec2').Image[],
 *   image?: import('@aws-sdk/client-ec2').Image,
 *   instanceTypes?: import('@aws-sdk/client-ec2').InstanceTypeInfo[],
 *   instanceId?: string,
 *   instanceIpAddress?: string,
 *   allocationId?: string,
 *   allocatedIpAddress?: string,
 *   associationId?: string,
 * }} State
 */

/**
 * A skip function provided to the `skipWhen` of a Step when you want
 * to ignore that step if any errors have occurred.
 * @param {State} state
 */
const skipWhenErrors = (state) => state.errors.length > 0;

const MAX_WAITER_TIME_IN_SECONDS = 60 * 8;

export const confirm = new ScenarioInput("confirmContinue", "Continue?", {
  type: "confirm",
  skipWhen: skipWhenErrors,
});

export const exitOnNoConfirm = new ScenarioAction(
  "exitOnConfirmContinueFalse",
  (/** @type { { earlyExit: boolean } & Record<string, any> } */ state) => {
    if (!state[confirm.name]) {
      state.earlyExit = true;
    }
  })
);

```

```

    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

export const greeting = new ScenarioOutput(
  "greeting",
  `
Welcome to the Amazon EC2 basic usage scenario.

Before you launch an instances, you'll need to provide a few things:
- A key pair - This is for SSH access to your EC2 instance. You only need to
  provide the name.
- A security group - This is used for configuring access to your instance.
  Again, only the name is needed.
- An IP address - Your public IP address will be fetched.
- An Amazon Machine Image (AMI)
- A compatible instance type`,
  { header: true, preformatted: true, skipWhen: skipWhenErrors },
);

export const provideKeyName = new ScenarioInput(
  "keyPairName",
  "Provide a name for a new key pair.",
  { type: "input", default: "ec2-example-key-pair", skipWhen: skipWhenErrors },
);

export const createKeyPair = new ScenarioAction(
  "createKeyPair",
  async (** @type {State} */ state) => {
    try {
      // Create a key pair in Amazon EC2.
      const { KeyMaterial, KeyPairId } = await state.ec2Client.send(
        // A unique name for the key pair. Up to 255 ASCII characters.
        new CreateKeyPairCommand({ KeyName: state[provideKeyName.name] }),
      );

      state.keyPairId = KeyPairId;

      // Save the private key in a temporary location.
      state.tmpDirectory = await mkdtemp(join(tmpdir(), "ec2-scenario-tmp"));
    }
  }
);

```



```

    await writeFile(
      `${state.tmpDirectory}/${state[provideKeyPairName.name]}.pem`,
      KeyMaterial,
      {
        mode: 0o400,
      },
    );
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidKeyPair.Duplicate"
    ) {
      caught.message = `${caught.message}. Try another key name.`;
    }

    state.errors.push(caught);
  },
  { skipWhen: skipWhenErrors },
);

export const logKeyPair = new ScenarioOutput(
  "logKeyPair",
  (/** @type {State} */ state) =>
    `Created the key pair ${state[provideKeyPairName.name]}.\`,
  { skipWhen: skipWhenErrors },
);

export const confirmDeleteKeyPair = new ScenarioInput(
  "confirmDeleteKeyPair",
  "Do you want to delete the key pair?",
  {
    type: "confirm",
    // Don't do anything when a key pair was never created.
    skipWhen: (/** @type {State} */ state) => !state.keyPairId,
  },
);

export const maybeDeleteKeyPair = new ScenarioAction(
  "deleteKeyPair",
  async (/** @type {State} */ state) => {
    try {
      // Delete a key pair by name from EC2
      await state.ec2Client.send(

```

```

        new DeleteKeyPairCommand({ KeyName: state[provideKeyPairName.name] })),
    );
} catch (caught) {
    if (
        caught instanceof Error &&
        // Occurs when a required parameter (e.g. KeyName) is undefined.
        caught.name === "MissingParameter"
    ) {
        caught.message = `${caught.message}. Did you provide the required value?`;
    }
    state.errors.push(caught);
}
},
{
    // Don't do anything when there's no key pair to delete or the user chooses
    // to keep it.
    skipWhen: (/** @type {State} */ state) =>
        !state.keyPairId || !state[confirmDeleteKeyPair.name],
},
);

export const provideSecurityGroupName = new ScenarioInput(
    "securityGroupName",
    "Provide a name for a new security group.",
    { type: "input", default: "ec2-scenario-sg", skipWhen: skipWhenErrors },
);

export const createSecurityGroup = new ScenarioAction(
    "createSecurityGroup",
    async (/** @type {State} */ state) => {
        try {
            // Create a new security group that will be used to configure ingress/
            // egress for
            // an EC2 instance.
            const { GroupId } = await state.ec2Client.send(
                new CreateSecurityGroupCommand({
                    GroupName: state[provideSecurityGroupName.name],
                    Description: "A security group for the Amazon EC2 example.",
                }),
            );
            state.securityGroupId = GroupId;
        } catch (caught) {
            if (caught instanceof Error && caught.name === "InvalidGroup.Duplicate") {

```

```

        caught.message = `${caught.message}. Please provide a different name for
your security group.`;
    }

    state.errors.push(caught);
  }
},
{ skipWhen: skipWhenErrors },
);

export const logSecurityGroup = new ScenarioOutput(
  "logSecurityGroup",
  (/** @type {State} */ state) =>
    `Created the security group ${state.securityGroupId}.`,
  { skipWhen: skipWhenErrors },
);

export const confirmDeleteSecurityGroup = new ScenarioInput(
  "confirmDeleteSecurityGroup",
  "Do you want to delete the security group?",
  {
    type: "confirm",
    // Don't do anything when a security group was never created.
    skipWhen: (/** @type {State} */ state) => !state.securityGroupId,
  },
);

export const maybeDeleteSecurityGroup = new ScenarioAction(
  "deleteSecurityGroup",
  async (/** @type {State} */ state) => {
    try {
      // Delete the security group if the 'skipWhen' condition below is not met.
      await state.ec2Client.send(
        new DeleteSecurityGroupCommand({
          GroupId: state.securityGroupId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidGroupId.Malformed"
      ) {
        caught.message = `${caught.message}. Please provide a valid GroupId.`;
      }
    }
  }
);

```

```

        state.errors.push(caught);
    }
},
{
    // Don't do anything when there's no security group to delete
    // or the user chooses to keep it.
    skipWhen: (** @type {State} */ state) =>
        !state.securityGroupId || !state[confirmDeleteSecurityGroup.name],
},
);

export const authorizeSecurityGroupIngress = new ScenarioAction(
    "authorizeSecurity",
    async (** @type {State} */ state) => {
        try {
            // Get the public IP address of the machine running this example.
            const ipAddress = await new Promise((res, rej) => {
                get("http://checkip.amazonaws.com", (response) => {
                    let data = "";
                    response.on("data", (chunk) => {
                        data += chunk;
                    });
                    response.on("end", () => res(data.trim()));
                }).on("error", (err) => {
                    rej(err);
                });
            });
            state.ipAddress = ipAddress;
            // Allow ingress from the IP address above to the security group.
            // This will allow you to SSH into the EC2 instance.
            const command = new AuthorizeSecurityGroupIngressCommand({
                GroupId: state.securityGroupId,
                IpPermissions: [
                    {
                        IpProtocol: "tcp",
                        FromPort: 22,
                        ToPort: 22,
                        IpRanges: [{ CidrIp: `${ipAddress}/32` }],
                    },
                ],
            });

            await state.ec2Client.send(command);
        } catch (caught) {

```

```
    if (
      caught instanceof Error &&
      caught.name === "InvalidGroupId.Malformed"
    ) {
      caught.message = `${caught.message}. Please provide a valid GroupId.`;
    }

    state.errors.push(caught);
  },
  { skipWhen: skipWhenErrors },
);

export const logSecurityGroupIngress = new ScenarioOutput(
  "logSecurityGroupIngress",
  (/** @type {State} */ state) =>
    `Allowed SSH access from your public IP: ${state.ipAddress}.`,
  { skipWhen: skipWhenErrors },
);

export const getImages = new ScenarioAction(
  "images",
  async (/** @type {State} */ state) => {
    const AMIs = [];
    // Some AWS services publish information about common artifacts as AWS
    Systems Manager (SSM)
    // public parameters. For example, the Amazon Elastic Compute Cloud (Amazon
    EC2)
    // service publishes information about Amazon Machine Images (AMIs) as public
    parameters.

    // Create the paginator for getting images. Actions that return multiple
    pages of
    // results have paginators to simplify those calls.
    const getParametersByPathPaginator = paginateGetParametersByPath(
      {
        // Not storing this client in state since it's only used once.
        client: new SSMClient({}),
      },
      {
        // The path to the public list of the latest amazon-linux instances.
        Path: "/aws/service/ami-amazon-linux-latest",
      },
    );
```

```

    try {
      for await (const page of getParametersByPathPaginator) {
        for (const param of page.Parameters) {
          // Filter by Amazon Linux 2
          if (param.Name.includes("amzn2")) {
            AMIs.push(param.Value);
          }
        }
      }
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidFilterValue") {
        caught.message = `${caught.message} Please provide a valid filter value
for paginateGetParametersByPath.`;
      }
      state.errors.push(caught);
      return;
    }

    const imageDetails = [];
    const describeImagesPaginator = paginateDescribeImages(
      { client: state.ec2Client },
      // The images found from the call to SSM.
      { ImageIds: AMIs },
    );

    try {
      // Get more details for the images found above.
      for await (const page of describeImagesPaginator) {
        imageDetails.push(...(page.Images || []));
      }

      // Store the image details for later use.
      state.images = imageDetails;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidAMIID.NotFound") {
        caught.message = `${caught.message}. Please provide a valid image id.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

```

```

export const provideImage = new ScenarioInput(
  "image",
  "Select one of the following images.",
  {
    type: "select",
    choices: (/** @type { State } */ state) =>
      state.images.map((image) => ({
        name: `${image.Description}`,
        value: image,
      })),
    default: (/** @type { State } */ state) => state.images[0],
    skipWhen: skipWhenErrors,
  },
);

export const getCompatibleInstanceTypes = new ScenarioAction(
  "getCompatibleInstanceTypes",
  async (/** @type {State} */ state) => {
    // Get more details about instance types that match the architecture of
    // the provided image.
    const paginator = paginateDescribeInstanceTypes(
      { client: state.ec2Client, pageSize: 25 },
      {
        Filters: [
          {
            Name: "processor-info.supported-architecture",
            // The value selected from provideImage()
            Values: [state.image.Architecture],
          },
          // Filter for smaller, less expensive, types.
          { Name: "instance-type", Values: ["*.micro", "/*.small"] },
        ],
      },
    );

    const instanceTypes = [];

    try {
      for await (const page of paginator) {
        if (page.InstanceTypes.length) {
          instanceTypes.push...(page.InstanceTypes || []);
        }
      }
    }
  }
);

```

```

    if (!instanceTypes.length) {
      state.errors.push(
        "No instance types matched the instance type filters.",
      );
    }
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      caught.message = `${caught.message}. Please check the provided values and
try again.`;
    }

    state.errors.push(caught);
  }

  state.instanceTypes = instanceTypes;
},
{ skipWhen: skipWhenErrors },
);

export const provideInstanceType = new ScenarioInput(
  "instanceType",
  "Select an instance type.",
  {
    choices: (/** @type {State} */ state) =>
      state.instanceTypes.map((instanceType) => ({
        name: `${instanceType.InstanceType} - Memory:
${instanceType.MemoryInfo.SizeInMiB}`,
        value: instanceType.InstanceType,
      })),
    type: "select",
    default: (/** @type {State} */ state) =>
      state.instanceTypes[0].InstanceType,
    skipWhen: skipWhenErrors,
  },
);

export const runInstance = new ScenarioAction(
  "runInstance",
  async (/** @type { State } */ state) => {
    const { Instances } = await state.ec2Client.send(
      new RunInstancesCommand({
        KeyName: state[provideKeyName.name],
        SecurityGroupIds: [state.securityGroupId],

```



```

        ImageId: state.image.ImageId,
        InstanceType: state[provideInstanceType.name],
        // Availability Zones have capacity limitations that may impact your
        ability to launch instances.
        // The `RunInstances` operation will only succeed if it can allocate at
        least the `MinCount` of instances.
        // However, EC2 will attempt to launch up to the `MaxCount` of instances,
        even if the full request cannot be satisfied.
        // If you need a specific number of instances, use `MinCount` and
        `MaxCount` set to the same value.
        // If you want to launch up to a certain number of instances, use
        `MaxCount` and let EC2 provision as many as possible.
        // If you require a minimum number of instances, but do not want to
        exceed a maximum, use both `MinCount` and `MaxCount`.
        MinCount: 1,
        MaxCount: 1,
    })),
);

state.instanceId = Instances[0].InstanceId;

try {
    // Poll `DescribeInstanceStatus` until status is "ok".
    await waitUntilInstanceStatusOk(
        {
            client: state.ec2Client,
            maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
        },
        { InstanceIds: [Instances[0].InstanceId] },
    );
} catch (caught) {
    if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in
the waiter.`;
    }

    state.errors.push(caught);
}
},
{ skipWhen: skipWhenErrors },
);

export const logRunInstance = new ScenarioOutput(
    "logRunInstance",

```

```

    "The next step is to run your EC2 instance for the first time. This can take a
    few minutes.",
    { header: true, skipWhen: skipWhenErrors },
  );

export const describeInstance = new ScenarioAction(
  "describeInstance",
  async (** @type { State } */ state) => {
    /** @type { import("@aws-sdk/client-ec2").Instance[] } */
    const instances = [];

    try {
      const paginator = paginateDescribeInstances(
        {
          client: state.ec2Client,
        },
        {
          // Only get our created instance.
          InstanceIds: [state.instanceId],
        },
      );

      for await (const page of paginator) {
        for (const reservation of page.Reservations) {
          instances.push(...reservation.Instances);
        }
      }

      if (instances.length !== 1) {
        throw new Error(`Instance ${state.instanceId} not found.`);
      }

      // The only info we need is the IP address for SSH purposes.
      state.instanceIpAddress = instances[0].PublicIpAddress;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "InvalidParameterValue") {
        caught.message = `${caught.message}. Please check provided values and try
        again.`;
      }

      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

```

```

export const logSSHConnectionInfo = new ScenarioOutput(
  "logSSHConnectionInfo",
  (/** @type { State } */ state) =>
    `You can now SSH into your instance using the following command:
ssh -i ${state.tmpDirectory}/${state[provideKeyPairName.name]}.pem ec2-user@
${state.instanceIpAddress}`,
  { preformatted: true, skipWhen: skipWhenErrors },
);

export const logStopInstance = new ScenarioOutput(
  "logStopInstance",
  "Stopping your EC2 instance.",
  { skipWhen: skipWhenErrors },
);

export const stopInstance = new ScenarioAction(
  "stopInstance",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new StopInstancesCommand({
          InstanceIds: [state.instanceId],
        }),
      );

      await waitUntilInstanceStopped(
        {
          client: state.ec2Client,
          maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
        },
        { InstanceIds: [state.instanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in
the waiter.`;
      }

      state.errors.push(caught);
    }
  },
  // Don't try to stop an instance that doesn't exist.
  { skipWhen: (/** @type { State } */ state) => !state.instanceId },

```

```
);

export const logIpAddressBehavior = new ScenarioOutput(
  "logIpAddressBehavior",
  [
    "When you run an instance, by default it's assigned an IP address.",
    "That IP address is not static. It will change every time the instance is restarted.",
    "The next step is to stop and restart your instance to demonstrate this behavior.",
  ].join(" "),
  { header: true, skipWhen: skipWhenErrors },
);

export const logStartInstance = new ScenarioOutput(
  "logStartInstance",
  (/** @type { State } */ state) => `Starting instance ${state.instanceId}`,
  { skipWhen: skipWhenErrors },
);

export const startInstance = new ScenarioAction(
  "startInstance",
  async (/** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new StartInstancesCommand({
          InstanceIds: [state.instanceId],
        }),
      );

      await waitUntilInstanceStatusOk(
        {
          client: state.ec2Client,
          maxWaitTime: MAX_WAITER_TIME_IN_SECONDS,
        },
        { InstanceIds: [state.instanceId] },
      );
    } catch (caught) {
      if (caught instanceof Error && caught.name === "TimeoutError") {
        caught.message = `${caught.message}. Try increasing the maxWaitTime in the waiter.`;
      }

      state.errors.push(caught);
    }
  }
);
```

```

    }
  },
  { skipWhen: skipWhenErrors },
);

export const logIpAllocation = new ScenarioOutput(
  "logIpAllocation",
  [
    "It is possible to have a static IP address.",
    "To demonstrate this, an IP will be allocated and associated to your EC2 instance.",
  ].join(" "),
  { header: true, skipWhen: skipWhenErrors },
);

export const allocateIp = new ScenarioAction(
  "allocateIp",
  async (** @type { State } */ state) => {
    try {
      // An Elastic IP address is allocated to your AWS account, and is yours until you release it.
      const { AllocationId, PublicIp } = await state.ec2Client.send(
        new AllocateAddressCommand({}),
      );
      state.allocationId = AllocationId;
      state.allocatedIpAddress = PublicIp;
    } catch (caught) {
      if (caught instanceof Error && caught.name === "MissingParameter") {
        caught.message = `${caught.message}. Did you provide these values?`;
      }
      state.errors.push(caught);
    }
  },
  { skipWhen: skipWhenErrors },
);

export const associateIp = new ScenarioAction(
  "associateIp",
  async (** @type { State } */ state) => {
    try {
      // Associate an allocated IP address to an EC2 instance. An IP address can be allocated
      // with the AllocateAddress action.
      const { AssociationId } = await state.ec2Client.send(

```

```

        new AssociateAddressCommand({
            AllocationId: state.allocationId,
            InstanceId: state.instanceId,
        })),
    );
    state.associationId = AssociationId;
    // Update the IP address that is being tracked to match
    // the one just associated.
    state.instanceIpAddress = state.allocatedIpAddress;
} catch (caught) {
    if (
        caught instanceof Error &&
        caught.name === "InvalidAllocationID.NotFound"
    ) {
        caught.message = `${caught.message}. Did you provide the ID of a valid
Elastic IP address AllocationId?`;
    }
    state.errors.push(caught);
}
},
{ skipWhen: skipWhenErrors },
);

export const logStaticIpProof = new ScenarioOutput(
    "logStaticIpProof",
    "The IP address should remain the same even after stopping and starting the
instance.",
    { header: true, skipWhen: skipWhenErrors },
);

export const logCleanUp = new ScenarioOutput(
    "logCleanUp",
    "That's it! You can choose to clean up the resources now, or clean them up on
your own later.",
    { header: true, skipWhen: skipWhenErrors },
);

export const confirmDisassociateAddress = new ScenarioInput(
    "confirmDisassociateAddress",
    "Do you want to disassociate and release the static IP address created
earlier?",
    {
        type: "confirm",
        skipWhen: (** @type { State } */ state) => !state.associationId,
    },
);

```

```

    },
  );

export const maybeDisassociateAddress = new ScenarioAction(
  "maybeDisassociateAddress",
  async (** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new DisassociateAddressCommand({
          AssociationId: state.associationId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidAssociationID.NotFound"
      ) {
        caught.message = `${caught.message}. Please provide a valid association
ID.`;
      }
      state.errors.push(caught);
    }
  },
  {
    skipWhen: (** @type { State } */ state) =>
      !state[confirmDisassociateAddress.name] || !state.associationId,
  },
);

export const maybeReleaseAddress = new ScenarioAction(
  "maybeReleaseAddress",
  async (** @type { State } */ state) => {
    try {
      await state.ec2Client.send(
        new ReleaseAddressCommand({
          AllocationId: state.allocationId,
        }),
      );
    } catch (caught) {
      if (
        caught instanceof Error &&
        caught.name === "InvalidAllocationID.NotFound"
      ) {

```

```

        caught.message = `${caught.message}. Please provide a valid
AllocationID.`;
    }
    state.errors.push(caught);
}
},
{
    skipWhen: (** @type { State } */ state) =>
        !state[confirmDisassociateAddress.name] || !state.allocationId,
},
);

export const confirmTerminateInstance = new ScenarioInput(
    "confirmTerminateInstance",
    "Do you want to terminate the instance?",
    // Don't do anything when an instance was never run.
    {
        skipWhen: (** @type { State } */ state) => !state.instanceId,
        type: "confirm",
    },
);

export const maybeTerminateInstance = new ScenarioAction(
    "terminateInstance",
    async (** @type { State } */ state) => {
        try {
            await state.ec2Client.send(
                new TerminateInstancesCommand({
                    InstanceIds: [state.instanceId],
                }),
            );
            await waitUntilInstanceTerminated(
                { client: state.ec2Client },
                { InstanceIds: [state.instanceId] },
            );
        } catch (caught) {
            if (caught instanceof Error && caught.name === "TimeoutError") {
                caught.message = `${caught.message}. Try increasing the maxWaitTime in
the waiter.`;
            }

            state.errors.push(caught);
        }
    },
);

```



```

    {
      // Don't do anything when there's no instance to terminate or the
      // user chooses not to terminate.
      skipWhen: (/** @type { State } */ state) =>
        !state.instanceId || !state[confirmTerminateInstance.name],
    },
  );

export const deleteTemporaryDirectory = new ScenarioAction(
  "deleteTemporaryDirectory",
  async (/** @type { State } */ state) => {
    try {
      await rm(state.tmpDirectory, { recursive: true });
    } catch (caught) {
      state.errors.push(caught);
    }
  },
);

export const logErrors = new ScenarioOutput(
  "logErrors",
  (/** @type {State} */ state) => {
    const errorList = state.errors
      .map((err) => ` - ${err.name}: ${err.message}`)
      .join("\n");
    return `Scenario errors found:\n${errorList}`;
  },
  {
    preformatted: true,
    header: true,
    // Don't log errors when there aren't any!
    skipWhen: (/** @type {State} */ state) => state.errors.length === 0,
  },
);

```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API Reference》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](#) 中查找完整示例，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
```

了解基本功能 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.terminateInstances(ti)
        ec2.waitForInstanceTerminated {
            // suspend call
            instanceIds = listOf(instanceIdVal)
        }
        println("$instanceIdVal is terminated!")
    }
}
```

```
    }  
}  
  
suspend fun releaseEC2AddressSc(allocId: String?) {  
    val request =  
        ReleaseAddressRequest {  
            allocationId = allocId  
        }  
  
    Ec2Client.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
minutes.")
        ec2.waitForInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}

suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
        println("Waiting until instance $instanceId stops. This will take a few
minutes.")
        ec2.waitForInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}
```

```

suspend fun describeEC2InstancesSc(newInstanceId: String?): String {
    var pubAddress = ""
    var isRunning = false
    val request =
        DescribeInstancesRequest {
            instanceIds = listOf(newInstanceId.toString())
        }

    while (!isRunning) {
        Ec2Client.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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 Reference》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

适用于 Python 的 SDK (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 EC2InstanceScenario:
    """
    A scenario that demonstrates how to use Boto3 to manage Amazon EC2 resources.
    Covers creating a key pair, security group, launching an instance,
    associating
    an Elastic IP, and cleaning up resources.
    """

    def __init__(
        self,
        inst_wrapper: EC2InstanceWrapper,
        key_wrapper: KeyPairWrapper,
        sg_wrapper: SecurityGroupWrapper,
        eip_wrapper: ElasticIpWrapper,
        ssm_client: boto3.client,
        remote_exec: bool = False,
    ):
        """
        Initializes the EC2InstanceScenario with the necessary AWS service
        wrappers.

        :param inst_wrapper: Wrapper for EC2 instance operations.
        :param key_wrapper: Wrapper for key pair operations.
        :param sg_wrapper: Wrapper for security group operations.
        :param eip_wrapper: Wrapper for Elastic IP operations.
        :param ssm_client: Boto3 client for accessing SSM to retrieve AMIs.
        :param remote_exec: Flag to indicate if the scenario is running in a
        remote execution
```



```

        environment. Defaults to False. If True, the script
won't prompt
        for user interaction.

        """
        self.inst_wrapper = inst_wrapper
        self.key_wrapper = key_wrapper
        self.sg_wrapper = sg_wrapper
        self.eip_wrapper = eip_wrapper
        self.ssm_client = ssm_client
        self.remote_exec = remote_exec

    def create_and_list_key_pairs(self) -> None:
        """
        Creates an RSA key pair for SSH access to the EC2 instance and lists
        available key pairs.
        """
        console.print("**Step 1: Create a Secure Key Pair**", style="bold cyan")
        console.print(
            "Let's create a secure RSA key pair for connecting to your EC2
instance."
        )
        key_name = f"MyUniqueKeyPair-{uuid.uuid4().hex[:8]}"
        console.print(f"- **Key Pair Name**: {key_name}")

        # Create the key pair and simulate the process with a progress bar.
        with alive_bar(1, title="Creating Key Pair") as bar:
            self.key_wrapper.create(key_name)
            time.sleep(0.4) # Simulate the delay in key creation
            bar()

        console.print(f"- **Private Key Saved to**:"
{self.key_wrapper.key_file_path}\n")

        # List key pairs (simulated) and show a progress bar.
        list_keys = True
        if list_keys:
            console.print("- Listing your key pairs...")
            start_time = time.time()
            with alive_bar(100, title="Listing Key Pairs") as bar:
                while time.time() - start_time < 2:
                    time.sleep(0.2)
                    bar(10)
                self.key_wrapper.list(5)
            if time.time() - start_time > 2:

```

```

        console.print(
            "Taking longer than expected! Please wait...",
            style="bold yellow",
        )

def create_security_group(self) -> None:
    """
    Creates a security group that controls access to the EC2 instance and
adds a rule
to allow SSH access from the user's current public IP address.
    """
    console.print("***Step 2: Create a Security Group**", style="bold cyan")
    console.print(
        "Security groups manage access to your instance. Let's create one."
    )
    sg_name = f"MySecurityGroup-{uuid.uuid4().hex[:8]}"
    console.print(f"- **Security Group Name**: {sg_name}")

    # Create the security group and simulate the process with a progress bar.
    with alive_bar(1, title="Creating Security Group") as bar:
        self.sg_wrapper.create(
            sg_name, "Security group for example: get started with
instances."
        )
        time.sleep(0.5)
        bar()

    console.print(f"- **Security Group ID**:
{self.sg_wrapper.security_group}\n")

    # Get the current public IP to set up SSH access.
    ip_response = urllib.request.urlopen("http://checkip.amazonaws.com")
    current_ip_address = ip_response.read().decode("utf-8").strip()
    console.print(
        "Let's add a rule to allow SSH only from your current IP address."
    )
    console.print(f"- **Your Public IP Address**: {current_ip_address}")
    console.print("- Automatically adding SSH rule...")

    # Update security group rules to allow SSH and simulate with a progress
bar.
    with alive_bar(1, title="Updating Security Group Rules") as bar:
        response = self.sg_wrapper.authorize_ingress(current_ip_address)
        time.sleep(0.4)

```

```

        if response and response.get("Return"):
            console.print("- **Security Group Rules Updated**.")
        else:
            console.print(
                "- **Error**: Couldn't update security group rules.",
                style="bold red",
            )
        bar()

    self.sg_wrapper.describe(self.sg_wrapper.security_group)

    def create_instance(self) -> None:
        """
        Launches an EC2 instance using an Amazon Linux 2 AMI and the created key
pair
and security group. Displays instance details and SSH connection
information.
        """
        # Retrieve Amazon Linux 2 AMIs from SSM.
        ami_paginator = self.ssm_client.get_paginator("get_parameters_by_path")
        ami_options = []
        for page in ami_paginator.paginate(Path="/aws/service/ami-amazon-linux-
latest"):
            ami_options += page["Parameters"]
        amzn2_images = self.inst_wrapper.get_images(
            [opt["Value"] for opt in ami_options if "amzn2" in opt["Name"]]
        )
        console.print("\n**Step 3: Launch Your Instance**", style="bold cyan")
        console.print(
            "Let's create an instance from an Amazon Linux 2 AMI. Here are some
options:"
        )
        image_choice = 0
        console.print(f"- Selected AMI: {amzn2_images[image_choice]
['ImageId']}\n")

        # Display instance types compatible with the selected AMI
        inst_types = self.inst_wrapper.get_instance_types(
            amzn2_images[image_choice]["Architecture"]
        )
        inst_type_choice = 0
        console.print(
            f"- Selected instance type: {inst_types[inst_type_choice]
['InstanceType']}\n"

```

```

    )

    console.print("Creating your instance and waiting for it to start...")
    with alive_bar(1, title="Creating Instance") as bar:
        self.inst_wrapper.create(
            amzn2_images[image_choice]["ImageId"],
            inst_types[inst_type_choice]["InstanceType"],
            self.key_wrapper.key_pair["KeyName"],
            [self.sg_wrapper.security_group],
        )
        time.sleep(21)
        bar()

    console.print(f"***Success! Your instance is ready:**\n", style="bold
green")
    self.inst_wrapper.display()

    console.print(
        "You can use SSH to connect to your instance. "
        "If the connection attempt times out, you might have to manually
update "
        "the SSH ingress rule for your IP address in the AWS Management
Console."
    )
    self._display_ssh_info()

    def _display_ssh_info(self) -> None:
        """
        Displays SSH connection information for the user to connect to the EC2
        instance.
        Handles the case where the instance does or does not have an associated
        public IP address.
        """
        if (
            not self.eip_wrapper.elastic_ips
            or not self.eip_wrapper.elastic_ips[0].allocation_id
        ):
            if self.inst_wrapper.instances:
                instance = self.inst_wrapper.instances[0]
                instance_id = instance["InstanceId"]

                waiter =
self.inst_wrapper.ec2_client.get_waiter("instance_running")
                console.print(

```

```

        "Waiting for the instance to be in a running state with a
public IP...",
        style="bold cyan",
    )

    with alive_bar(1, title="Waiting for Instance to Start") as bar:
        waiter.wait(InstanceIds=[instance_id])
        time.sleep(20)
        bar()

    instance = self.inst_wrapper.ec2_client.describe_instances(
        InstanceIds=[instance_id]
    )["Reservations"][0]["Instances"][0]

    public_ip = instance.get("PublicIpAddress")
    if public_ip:
        console.print(
            "\nTo connect via SSH, open another command prompt and
run the following command:",
            style="bold cyan",
        )
        console.print(
            f"\tssh -i {self.key_wrapper.key_file_path} ec2-
user@{public_ip}"
        )
    else:
        console.print(
            "Instance does not have a public IP address assigned.",
            style="bold red",
        )
    else:
        console.print(
            "No instance available to retrieve public IP address.",
            style="bold red",
        )
    else:
        elastic_ip = self.eip_wrapper.elastic_ips[0]
        elastic_ip_address = elastic_ip.public_ip
        console.print(
            f"\tssh -i {self.key_wrapper.key_file_path} ec2-
user@{elastic_ip_address}"
        )

    if not self.remote_exec:

```

```

        console.print("\nOpen a new terminal tab to try the above SSH
command.")
        input("Press Enter to continue...")

    def associate_elastic_ip(self) -> None:
        """
        Allocates an Elastic IP address and associates it with the EC2 instance.
        Displays the Elastic IP address and SSH connection information.
        """
        console.print("\n**Step 4: Allocate an Elastic IP Address**", style="bold
cyan")
        console.print(
            "You can allocate an Elastic IP address and associate it with your
instance\n"
            "to keep a consistent IP address even when your instance restarts."
        )

        with alive_bar(1, title="Allocating Elastic IP") as bar:
            elastic_ip = self.eip_wrapper.allocate()
            time.sleep(0.5)
            bar()

        console.print(
            f"- **Allocated Static Elastic IP Address**: {elastic_ip.public_ip}."
        )

        with alive_bar(1, title="Associating Elastic IP") as bar:
            self.eip_wrapper.associate(
                elastic_ip.allocation_id, self.inst_wrapper.instances[0]
["InstanceId"]
            )
            time.sleep(2)
            bar()

        console.print(f"- **Associated Elastic IP with Your Instance**.")
        console.print(
            "You can now use SSH to connect to your instance by using the Elastic
IP."
        )
        self._display_ssh_info()

    def stop_and_start_instance(self) -> None:
        """
        Stops and restarts the EC2 instance. Displays instance state and explains

```

```

        changes that occur when the instance is restarted, such as the potential
change
        in the public IP address unless an Elastic IP is associated.
        """
        console.print("\n**Step 5: Stop and Start Your Instance**", style="bold
cyan")
        console.print("Let's stop and start your instance to see what changes.")
        console.print("- **Stopping your instance and waiting until it's
stopped...**")

        with alive_bar(1, title="Stopping Instance") as bar:
            self.inst_wrapper.stop()
            time.sleep(360)
            bar()

        console.print("- **Your instance is stopped. Restarting...**")

        with alive_bar(1, title="Starting Instance") as bar:
            self.inst_wrapper.start()
            time.sleep(20)
            bar()

        console.print("**Your instance is running.**", style="bold green")
        self.inst_wrapper.display()

        elastic_ip = (
            self.eip_wrapper.elastic_ips[0] if self.eip_wrapper.elastic_ips else
None
        )

        if elastic_ip is None or elastic_ip.allocation_id is None:
            console.print(
                "- **Note**: Every time your instance is restarted, its public IP
address changes."
            )
        else:
            console.print(
                f"Because you have associated an Elastic IP with your instance,
you can \n"
                f"connect by using a consistent IP address after the instance
restarts: {elastic_ip.public_ip}"
            )

        self._display_ssh_info()

```

```

def cleanup(self) -> None:
    """
    Cleans up all the resources created during the scenario, including
    disassociating
    and releasing the Elastic IP, terminating the instance, deleting the
    security
    group, and deleting the key pair.
    """
    console.print("\n**Step 6: Clean Up Resources**", style="bold cyan")
    console.print("Cleaning up resources:")

    for elastic_ip in self.eip_wrapper.elastic_ips:
        console.print(f"- **Elastic IP**: {elastic_ip.public_ip}")

        with alive_bar(1, title="Disassociating Elastic IP") as bar:
            self.eip_wrapper.disassociate(elastic_ip.allocation_id)
            time.sleep(2)
            bar()

        console.print("\t- **Disassociated Elastic IP from the Instance**")

        with alive_bar(1, title="Releasing Elastic IP") as bar:
            self.eip_wrapper.release(elastic_ip.allocation_id)
            time.sleep(1)
            bar()

        console.print("\t- **Released Elastic IP**")

    console.print(f"- **Instance**: {self.inst_wrapper.instances[0]
['InstanceId']}")

    with alive_bar(1, title="Terminating Instance") as bar:
        self.inst_wrapper.terminate()
        time.sleep(380)
        bar()

    console.print("\t- **Terminated Instance**")

    console.print(f"- **Security Group**: {self.sg_wrapper.security_group}")

    with alive_bar(1, title="Deleting Security Group") as bar:
        self.sg_wrapper.delete(self.sg_wrapper.security_group)
        time.sleep(1)

```



```

        bar()

        console.print("\t- **Deleted Security Group**")

        console.print(f"- **Key Pair**: {self.key_wrapper.key_pair['KeyName']}")

        with alive_bar(1, title="Deleting Key Pair") as bar:
            self.key_wrapper.delete(self.key_wrapper.key_pair["KeyName"])
            time.sleep(0.4)
            bar()

        console.print("\t- **Deleted Key Pair**")

    def run_scenario(self) -> None:
        """
        Executes the entire EC2 instance scenario: creates key pairs, security
        groups,
        launches an instance, associates an Elastic IP, and cleans up all
        resources.
        """
        logging.basicConfig(level=logging.INFO, format="%(levelname)s:
%(message)s")

        console.print("-" * 88)
        console.print(
            "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started
            with instances demo.",
            style="bold magenta",
        )
        console.print("-" * 88)

        self.create_and_list_key_pairs()
        self.create_security_group()
        self.create_instance()
        self.stop_and_start_instance()
        self.associate_elastic_ip()
        self.stop_and_start_instance()
        self.cleanup()

        console.print("\nThanks for watching!", style="bold green")
        console.print("-" * 88)

if __name__ == "__main__":

```

```

try:
    scenario = EC2InstanceScenario(
        EC2InstanceWrapper.from_client(),
        KeyPairWrapper.from_client(),
        SecurityGroupWrapper.from_client(),
        ElasticIpWrapper.from_client(),
        boto3.client("ssm"),
    )
    scenario.run_scenario()
except Exception:
    logging.exception("Something went wrong with the demo.")

```

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.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
        directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is
        stored.
                           This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                           This is a high-level object that wraps key pair actions.
        Optional.
        """
        self.ec2_client = ec2_client

```

```

        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2
client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())

    def create(self, key_name: str) -> dict:
        """
        Creates a key pair that can be used to securely connect to an EC2
instance.
        The returned key pair contains private key information that cannot be
retrieved
        again. The private key data is stored as a .pem file.

        :param key_name: The name of the key pair to create.
        :return: A dictionary representing the Boto3 KeyPair object that
represents the newly created key pair.
        :raises ClientError: If there is an error in creating the key pair, for
example, if a key pair with the same name already exists.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_name)
            self.key_pair = response
            self.key_file_path = os.path.join(
                self.key_file_dir.name, f"{self.key_pair['KeyName']}.pem"
            )
            with open(self.key_file_path, "w") as key_file:
                key_file.write(self.key_pair["KeyMaterial"])
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidKeyPair.Duplicate":
                logger.error(
                    f"A key pair called {key_name} already exists. "
                    "Please choose a different name for your key pair "

```

```

        "or delete the existing key pair before creating."
    )
    raise
else:
    return self.key_pair

def list(self, limit: Optional[int] = None) -> None:
    """
    Displays a list of key pairs for the current account.

    WARNING: Results are not paginated.

    :param limit: The maximum number of key pairs to list. If not specified,
                  all key pairs will be listed.
    :raises ClientError: If there is an error in listing the key pairs.
    """
    try:
        response = self.ec2_client.describe_key_pairs()
        key_pairs = response.get("KeyPairs", [])

        if limit:
            key_pairs = key_pairs[:limit]

        for key_pair in key_pairs:
            logger.info(
                f"Found {key_pair['KeyType']} key '{key_pair['KeyName']}'
with fingerprint:"
            )
            logger.info(f"\t{key_pair['KeyFingerprint']}")
    except ClientError as err:
        logger.error(f"Failed to list key pairs: {str(err)}")
        raise

def delete(self, key_name: str) -> bool:
    """
    Deletes a key pair by its name.

    :param key_name: The name of the key pair to delete.
    :return: A boolean indicating whether the deletion was successful.
    :raises ClientError: If there is an error in deleting the key pair, for
example,
                        if the key pair does not exist.
    """

```

```

        """
    try:
        self.ec2_client.delete_key_pair(KeyName=key_name)
        logger.info(f"Successfully deleted key pair: {key_name}")
        self.key_pair = None
        return True
    except self.ec2_client.exceptions.ClientError as err:
        logger.error(f"Deletion failed for key pair: {key_name}")
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidKeyPair.NotFound":
            logger.error(
                f"The key pair '{key_name}' does not exist and cannot be
deleted. "
                "Please verify the key pair name and try again."
            )
        raise

```

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_client: boto3.client, security_group: Optional[str] =
None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                        access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
high-level identifier
                        that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

```

```

@classmethod
def from_client(cls) -> "SecurityGroupWrapper":
    """
    Creates a SecurityGroupWrapper instance with a default EC2 client.

    :return: An instance of SecurityGroupWrapper initialized with the default
    EC2 client.
    """
    ec2_client = boto3.client("ec2")
    return cls(ec2_client)

def create(self, group_name: str, group_description: str) -> str:
    """
    Creates a security group in the default virtual private cloud (VPC) of
    the current account.

    :param group_name: The name of the security group to create.
    :param group_description: The description of the security group to
    create.
    :return: The ID of the newly created security group.
    :raise Handles AWS SDK service-level ClientError, with special handling
    for ResourceAlreadyExists
    """
    try:
        response = self.ec2_client.create_security_group(
            GroupName=group_name, Description=group_description
        )
        self.security_group = response["GroupId"]
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceAlreadyExists":
            logger.error(
                f"Security group '{group_name}' already exists. Please choose
                a different name."
            )
            raise
        else:
            return self.security_group

def authorize_ingress(self, ssh_ingress_ip: str) -> Optional[Dict[str, Any]]:
    """
    Adds a rule to the security group to allow access to SSH.

```

```

        :param ssh_ingress_ip: The IP address that is granted inbound access to
        connect
                                to port 22 over TCP, used for SSH.
        :return: The response to the authorization request. The 'Return' field of
        the
                response indicates whether the request succeeded or failed, or
        None if no security group is set.
        :raise Handles AWS SDK service-level ClientError, with special handling
        for ResourceAlreadyExists
        """
        if self.security_group is None:
            logger.info("No security group to update.")
            return None

        try:
            ip_permissions = [
                {
                    # SSH ingress open to only the specified IP address.
                    "IpProtocol": "tcp",
                    "FromPort": 22,
                    "ToPort": 22,
                    "IpRanges": [{"CidrIp": f"{ssh_ingress_ip}/32"}],
                }
            ]
            response = self.ec2_client.authorize_security_group_ingress(
                GroupId=self.security_group, IpPermissions=ip_permissions
            )
        except ClientError as err:
            if err.response["Error"]["Code"] == "InvalidPermission.Duplicate":
                logger.error(
                    f"The SSH ingress rule for IP {ssh_ingress_ip} already
exists"
                    f"in security group '{self.security_group}'."
                )
                raise
            else:
                return response

    def describe(self, security_group_id: Optional[str] = None) -> bool:
        """
        Displays information about the specified security group or all security
        groups if no ID is provided.

```

```

        :param security_group_id: The ID of the security group to describe.
                                   If None, an open search is performed to
describe all security groups.
        :returns: True if the description is successful.
        :raises ClientError: If there is an error describing the security
group(s), such as an invalid security group ID.
    """
    try:
        paginator = self.ec2_client.get_paginator("describe_security_groups")

        if security_group_id is None:
            # If no ID is provided, return all security groups.
            page_iterator = paginator.paginate()
        else:
            page_iterator = paginator.paginate(GroupIds=[security_group_id])

        for page in page_iterator:
            for security_group in page["SecurityGroups"]:
                print(f"Security group: {security_group['GroupName']}")
                print(f"\tID: {security_group['GroupId']}")
                print(f"\tVPC: {security_group['VpcId']}")
                if security_group["IpPermissions"]:
                    print("Inbound permissions:")
                    pp(security_group["IpPermissions"])

        return True
    except ClientError as err:
        logger.error("Failed to describe security group(s).")
        if err.response["Error"]["Code"] == "InvalidGroup.NotFound":
            logger.error(
                f"Security group {security_group_id} does not exist "
                f"because the specified security group ID was not found."
            )
        raise

def delete(self, security_group_id: str) -> bool:
    """
    Deletes the specified security group.

    :param security_group_id: The ID of the security group to delete.
Required.

    :returns: True if the deletion is successful.

```



```

        :raises ClientError: If the security group cannot be deleted due to an
        AWS service error.
        """
        try:
            self.ec2_client.delete_security_group(GroupId=security_group_id)
            logger.info(f"Successfully deleted security group
            '{security_group_id}'")
            return True
        except ClientError as err:
            logger.error(f"Deletion failed for security group
            '{security_group_id}'")
            error_code = err.response["Error"]["Code"]

            if error_code == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted
            because it does not exist."
                )
            elif error_code == "DependencyViolation":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted
            because it is still in use."
                    " Verify that it is:"
                    "\n\t- Detached from resources"
                    "\n\t- Removed from references in other groups"
                    "\n\t- Removed from VPC's as a default group"
                )
            raise

```

定义一个包装实例操作的类。

```

class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """

```

```

        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def create(
        self,
        image_id: str,
        instance_type: str,
        key_pair_name: str,
        security_group_ids: Optional[List[str]] = None,
    ) -> List[Dict[str, Any]]:
        """
        Creates a new EC2 instance in the default VPC of the current account.

        The instance starts immediately after it is created.

        :param image_id: The ID of the Amazon Machine Image (AMI) to use for the
instance.
        :param instance_type: The type of instance to create, such as 't2.micro'.
        :param key_pair_name: The name of the key pair to use for SSH access.
        :param security_group_ids: A list of security group IDs to associate with
the instance.

```

```

                                If not specified, the default security group
of the VPC is used.
    :return: A list of dictionaries representing Boto3 Instance objects
representing the newly created instances.
    """
    try:
        instance_params = {
            "ImageId": image_id,
            "InstanceType": instance_type,
            "KeyName": key_pair_name,
        }
        if security_group_ids is not None:
            instance_params["SecurityGroupIds"] = security_group_ids

        response = self.ec2_client.run_instances(
            **instance_params, MinCount=1, MaxCount=1
        )
        instance = response["Instances"][0]
        self.instances.append(instance)
        waiter = self.ec2_client.get_waiter("instance_running")
        waiter.wait(InstanceIds=[instance["InstanceId"]])
    except ClientError as err:
        params_str = "\n\t".join(
            f"{key}: {value}" for key, value in instance_params.items()
        )
        logger.error(
            f"Failed to complete instance creation request.\nRequest details:
{params_str}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "InstanceLimitExceeded":
            logger.error(
                (
                    f"Insufficient capacity for instance type
'{instance_type}'. "
                    "Terminate unused instances or contact AWS Support for a
limit increase."
                )
            )
        if error_code == "InsufficientInstanceCapacity":
            logger.error(
                (
                    f"Insufficient capacity for instance type
'{instance_type}'. "

```

```

        "Select a different instance type or launch in a
different availability zone."
    )
    )
    raise
return self.instances

def display(self, state_filter: Optional[str] = "running") -> None:
    """
    Displays information about instances, filtering by the specified state.

    :param state_filter: The instance state to include in the output. Only
instances in this state
                        will be displayed. Default is 'running'. Example
states: 'running', 'stopped'.
    """
    if not self.instances:
        logger.info("No instances to display.")
        return

    instance_ids = [instance["InstanceId"] for instance in self.instances]
    paginator = self.ec2_client.get_paginator("describe_instances")
    page_iterator = paginator.paginate(InstanceIds=instance_ids)

    try:
        for page in page_iterator:
            for reservation in page["Reservations"]:
                for instance in reservation["Instances"]:
                    instance_state = instance["State"]["Name"]

                    # Apply the state filter (default is 'running')
                    if state_filter and instance_state != state_filter:
                        continue # Skip this instance if it doesn't match
the filter

                    # Create a formatted string with instance details
                    instance_info = (
                        f"• ID: {instance['InstanceId']}\n"
                        f"• Image ID: {instance['ImageId']}\n"
                        f"• Instance type: {instance['InstanceType']}\n"
                        f"• Key name: {instance['KeyName']}\n"
                        f"• VPC ID: {instance['VpcId']}\n"

```

```

        f"• Public IP: {instance.get('PublicIpAddress', 'N/A')}\n"

        f"• State: {instance_state}"
    )
    print(instance_info)

except ClientError as err:
    logger.error(
        f"Failed to display instance(s). : {' '.join(map(str, instance_ids))}"
    )
    error_code = err.response["Error"]["Code"]
    if error_code == "InvalidInstanceID.NotFound":
        logger.error(
            "One or more instance IDs do not exist. "
            "Please verify the instance IDs and try again."
        )
        raise

def terminate(self) -> None:
    """
    Terminates instances and waits for them to reach the terminated state.
    """
    if not self.instances:
        logger.info("No instances to terminate.")
        return

    instance_ids = [instance["InstanceId"] for instance in self.instances]
    try:
        self.ec2_client.terminate_instances(InstanceIds=instance_ids)
        waiter = self.ec2_client.get_waiter("instance_terminated")
        waiter.wait(InstanceIds=instance_ids)
        self.instances.clear()
        for instance_id in instance_ids:
            print(f"• Instance ID: {instance_id}\n" f"• Action: Terminated")

    except ClientError as err:
        logger.error(
            f"Failed instance termination details:\n\t{str(self.instances)}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidInstanceID.NotFound":
            logger.error(

```

```

        "One or more instance IDs do not exist. "
        "Please verify the instance IDs and try again."
    )
    raise

def start(self) -> Optional[Dict[str, Any]]:
    """
    Starts instances and waits for them to be in a running state.

    :return: The response to the start request.
    """
    if not self.instances:
        logger.info("No instances to start.")
        return None

    instance_ids = [instance["InstanceId"] for instance in self.instances]
    try:
        start_response =
self.ec2_client.start_instances(InstanceIds=instance_ids)
        waiter = self.ec2_client.get_waiter("instance_running")
        waiter.wait(InstanceIds=instance_ids)
        return start_response
    except ClientError as err:
        logger.error(
            f"Failed to start instance(s): {' '.join(map(str,
instance_ids))}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "IncorrectInstanceState":
            logger.error(
                "Couldn't start instance(s) because they are in an incorrect
state. "
                "Ensure the instances are in a stopped state before starting
them."
            )
        raise

def stop(self) -> Optional[Dict[str, Any]]:
    """
    Stops instances and waits for them to be in a stopped state.

```

```

        :return: The response to the stop request, or None if there are no
instances to stop.
        """
        if not self.instances:
            logger.info("No instances to stop.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            # Attempt to stop the instances
            stop_response =
self.ec2_client.stop_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_stopped")
            waiter.wait(InstanceIds=instance_ids)
        except ClientError as err:
            logger.error(
                f"Failed to stop instance(s): {' '.join(map(str, instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't stop instance(s) because they are in an incorrect
state. "
                    "Ensure the instances are in a running state before stopping
them."
                )
            raise
        return stop_response

def get_images(self, image_ids: List[str]) -> List[Dict[str, Any]]:
    """
    Gets information about Amazon Machine Images (AMIs) from a list of AMI
IDs.

    :param image_ids: The list of AMI IDs to look up.
    :return: A list of dictionaries representing the requested AMIs.
    """
    try:
        response = self.ec2_client.describe_images(ImageIds=image_ids)
        images = response["Images"]
    except ClientError as err:
        logger.error(f"Failed to stop AMI(s): {' '.join(map(str,
image_ids))}")

```

```

        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidAMIID.NotFound":
            logger.error("One or more of the AMI IDs does not exist.")
            raise
    return images

def get_instance_types(
    self, architecture: str = "x86_64", sizes: List[str] = ["*.micro",
    "*.small"]
) -> List[Dict[str, Any]]:
    """
    Gets instance types that support the specified architecture and size.
    See https://docs.aws.amazon.com/AWSEC2/latest/APIReference/
    API_DescribeInstanceTypes.html
    for a list of allowable parameters.

    :param architecture: The architecture supported by instance types.
    Default: 'x86_64'.
    :param sizes: The size of instance types. Default: '*.micro', '*.small',
    :return: A list of dictionaries representing instance types that support
    the specified architecture and size.
    """
    try:
        inst_types = []
        paginator = self.ec2_client.get_paginator("describe_instance_types")
        for page in paginator.paginate(
            Filters=[
                {
                    "Name": "processor-info.supported-architecture",
                    "Values": [architecture],
                },
                {"Name": "instance-type", "Values": sizes},
            ]
        ):
            inst_types += page["InstanceTypes"]
    except ClientError as err:
        logger.error(
            f"Failed to get instance types: {architecture},
            {'', '.join(map(str, sizes))}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidParameterValue":
            logger.error(

```



```

        "Parameters are invalid. "
        "Ensure architecture and size strings conform to
DescribeInstanceTypes API reference."
    )
    raise
else:
    return inst_types

```

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 using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
            = None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

        def __init__(self, ec2_client: Any) -> None:
            """
            Initializes the ElasticIpWrapper with an EC2 client.

            :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                                access to AWS EC2 services.

```

```

        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def allocate(self) -> "ElasticIpWrapper.ElasticIp":
        """
        Allocates an Elastic IP address that can be associated with an Amazon EC2
instance. By using an Elastic IP address, you can keep the public IP
address
constant even when you restart the associated instance.

        :return: The ElasticIp object for the newly created Elastic IP address.
        :raises ClientError: If the allocation fails, such as reaching the
maximum limit of Elastic IPs.
        """
        try:
            response = self.ec2_client.allocate_address(Domain="vpc")
            elastic_ip = self.ElasticIp(
                allocation_id=response["AllocationId"],
                public_ip=response["PublicIp"]
            )
            self.elastic_ips.append(elastic_ip)
        except ClientError as err:
            if err.response["Error"]["Code"] == "AddressLimitExceeded":
                logger.error(
                    "Max IP's reached. Release unused addresses or contact AWS
Support for an increase."
                )
            raise err
        return elastic_ip

```

```

def associate(
    self, allocation_id: str, instance_id: str
) -> Union[Dict[str, Any], None]:
    """
    Associates an Elastic IP address with an instance. When this association
    is created, the Elastic IP's public IP address is immediately used as the
    public IP address of the associated instance.

    :param allocation_id: The allocation ID of the Elastic IP.
    :param instance_id: The ID of the Amazon EC2 instance.
    :return: A response that contains the ID of the association, or None if
    no Elastic IP is found.
    :raises ClientError: If the association fails, such as when the instance
    ID is not found.
    """
    elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
    if elastic_ip is None:
        logger.info(f"No Elastic IP found with allocation ID
{allocation_id}.")
        return None

    try:
        response = self.ec2_client.associate_address(
            AllocationId=allocation_id, InstanceId=instance_id
        )
        elastic_ip.instance_id = (
            instance_id # Track the instance associated with this Elastic
IP.
        )
    except ClientError as err:
        if err.response["Error"]["Code"] == "InvalidInstanceId.NotFound":
            logger.error(
                f"Failed to associate Elastic IP {allocation_id} with
{instance_id} "
                "because the specified instance ID does not exist or has not
propagated fully. "
                "Verify the instance ID and try again, or wait a few moments
before attempting to "
                "associate the Elastic IP address."
            )
            raise

```

```

    return response

def disassociate(self, allocation_id: str) -> None:
    """
    Removes an association between an Elastic IP address and an instance.
    When the
    association is removed, the instance is assigned a new public IP address.

    :param allocation_id: The allocation ID of the Elastic IP to
    disassociate.
    :raises ClientError: If the disassociation fails, such as when the
    association ID is not found.
    """
    elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
    allocation_id)
    if elastic_ip is None or elastic_ip.instance_id is None:
        logger.info(
            f"No association found for Elastic IP with allocation ID
    {allocation_id}."
        )
        return

    try:
        # Retrieve the association ID before disassociating
        response =
    self.ec2_client.describe_addresses(AllocationIds=[allocation_id])
        association_id = response["Addresses"][0].get("AssociationId")

        if association_id:

    self.ec2_client.disassociate_address(AssociationId=association_id)
            elastic_ip.instance_id = None # Remove the instance association
        else:
            logger.info(
                f"No Association ID found for Elastic IP with allocation ID
    {allocation_id}."
            )

    except ClientError as err:
        if err.response["Error"]["Code"] == "InvalidAssociationID.NotFound":
            logger.error(
                f"Failed to disassociate Elastic IP {allocation_id} "

```

```

        "because the specified association ID for the Elastic IP
address was not found. "
        "Verify the association ID and ensure the Elastic IP is
currently associated with a "
        "resource before attempting to disassociate it."
    )
    raise

def release(self, allocation_id: str) -> None:
    """
    Releases an Elastic IP address. After the Elastic IP address is released,
    it can no longer be used.

    :param allocation_id: The allocation ID of the Elastic IP to release.
    :raises ClientError: If the release fails, such as when the Elastic IP
address is not found.
    """
    elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
    if elastic_ip is None:
        logger.info(f"No Elastic IP found with allocation ID
{allocation_id}.")
        return

    try:
        self.ec2_client.release_address(AllocationId=allocation_id)
        self.elastic_ips.remove(elastic_ip) # Remove the Elastic IP from the
list
    except ClientError as err:
        if err.response["Error"]["Code"] == "InvalidAddress.NotFound":
            logger.error(
                f"Failed to release Elastic IP address {allocation_id} "
                "because it could not be found. Verify the Elastic IP address
"
                "and ensure it is allocated to your account in the correct
region "
                "before attempting to release it."
            )
            raise

    @staticmethod
    def get_elastic_ip_by_allocation(

```

```

        elastic_ips: List["ElasticIpWrapper.ElasticIp"], allocation_id: str
    ) -> Optional["ElasticIpWrapper.ElasticIp"]:
        """
        Retrieves an Elastic IP object by its allocation ID from a given list of
        Elastic IPs.

        :param elastic_ips: A list of ElasticIp objects.
        :param allocation_id: The allocation ID of the Elastic IP to retrieve.
        :return: The ElasticIp object associated with the allocation ID, or None
        if not found.
        """
        return next(
            (ip for ip in elastic_ips if ip.allocation_id == allocation_id), None
        )

```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》中的以下主题。

- [AllocateAddress](#)
- [AssociateAddress](#)
- [AuthorizeSecurityGroupIngress](#)
- [CreateKeyPair](#)
- [CreateSecurityGroup](#)
- [DeleteKeyPair](#)
- [DeleteSecurityGroup](#)
- [DescribeImages](#)
- [DescribeInstanceTypes](#)
- [DescribeInstances](#)
- [DescribeKeyPairs](#)
- [DescribeSecurityGroups](#)
- [DisassociateAddress](#)
- [ReleaseAddress](#)
- [RunInstances](#)
- [StartInstances](#)

- [TerminateInstances](#)
- [UnmonitorInstances](#)

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](#) 中查找完整示例，了解如何进行设置和运行。

该 `EC2InstanceScenario` 实现包含将示例作为一个整体运行的逻辑。

```
//! Scenario that uses the AWS SDK for Rust (the SDK) with Amazon Elastic Compute
    Cloud
//! (Amazon EC2) to do the following:
//!
//! * Create a key pair that is used to secure SSH communication between your
    computer and
//!   an EC2 instance.
//! * Create a security group that acts as a virtual firewall for your EC2
    instances to
//!   control incoming and outgoing traffic.
//! * Find an Amazon Machine Image (AMI) and a compatible instance type.
//! * Create an instance that is created from the instance type and AMI you
    select, and
//!   is configured to use the security group and key pair created in this
    example.
//! * Stop and restart the instance.
//! * Create an Elastic IP address and associate it as a consistent IP address
    for your instance.
//! * Connect to your instance with SSH, using both its public IP address and
    your Elastic IP
    address.
//! * Clean up all of the resources created by this example.

use std::net::Ipv4Addr;
```

```

use crate::{
    ec2::{EC2Error, EC2},
    getting_started::{key_pair::KeyPairManager, util::Util},
    ssm::SSM,
};
use aws_sdk_ssm::types::Parameter;

use super::{
    elastic_ip::ElasticIpManager, instance::InstanceManager,
    security_group::SecurityGroupManager,
    util::ScenarioImage,
};

pub struct Ec2InstanceScenario {
    ec2: EC2,
    ssm: SSM,
    util: Util,
    key_pair_manager: KeyPairManager,
    security_group_manager: SecurityGroupManager,
    instance_manager: InstanceManager,
    elastic_ip_manager: ElasticIpManager,
}

impl Ec2InstanceScenario {
    pub fn new(ec2: EC2, ssm: SSM, util: Util) -> Self {
        Ec2InstanceScenario {
            ec2,
            ssm,
            util,
            key_pair_manager: Default::default(),
            security_group_manager: Default::default(),
            instance_manager: Default::default(),
            elastic_ip_manager: Default::default(),
        }
    }

    pub async fn run(&mut self) -> Result<(), EC2Error> {
        self.create_and_list_key_pairs().await?;
        self.create_security_group().await?;
        self.create_instance().await?;
        self.stop_and_start_instance().await?;
        self.associate_elastic_ip().await?;
        self.stop_and_start_instance().await?;
        Ok(())
    }
}

```



```

    }

    /// 1. Creates an RSA key pair and saves its private key data as a .pem file
    in secure
    ///    temporary storage. The private key data is deleted after the example
    completes.
    /// 2. Optionally, lists the first five key pairs for the current account.
    pub async fn create_and_list_key_pairs(&mut self) -> Result<(), EC2Error> {
        println!( "Let's create an RSA key pair that you can be use to securely
        connect to your EC2 instance.");

        let key_name = self.util.prompt_key_name()?;

        self.key_pair_manager
            .create(&self.ec2, &self.util, key_name)
            .await?;

        println!(
            "Created a key pair {} and saved the private key to {:?}.",
            self.key_pair_manager
                .key_pair()
                .key_name()
                .ok_or_else(|| EC2Error::new("No key name after creating key"))?,
            self.key_pair_manager
                .key_file_path()
                .ok_or_else(|| EC2Error::new("No key file after creating key"))?
        );

        if self.util.should_list_key_pairs()? {
            for pair in self.key_pair_manager.list(&self.ec2).await? {
                println!(
                    "Found {:?} key {} with fingerprint:\t{:?}",
                    pair.key_type(),
                    pair.key_name().unwrap_or("Unknown"),
                    pair.key_fingerprint()
                );
            }
        }

        Ok(())
    }

    /// 1. Creates a security group for the default VPC.
    /// 2. Adds an inbound rule to allow SSH. The SSH rule allows only

```

```

    /// inbound traffic from the current computer's public IPv4 address.
    /// 3. Displays information about the security group.
    ///
    /// This function uses <http://checkip.amazonaws.com> to get the current
    public IP
    /// address of the computer that is running the example. This method works in
    most
    /// cases. However, depending on how your computer connects to the internet,
    you
    /// might have to manually add your public IP address to the security group
    by using
    /// the AWS Management Console.
    pub async fn create_security_group(&mut self) -> Result<(), EC2Error> {
        println!("Let's create a security group to manage access to your
        instance.");
        let group_name = self.util.prompt_security_group_name()?;

        self.security_group_manager
            .create(
                &self.ec2,
                &group_name,
                "Security group for example: get started with instances.",
            )
            .await?;

        println!(
            "Created security group {} in your default VPC {}.",
            self.security_group_manager.group_name(),
            self.security_group_manager
                .vpc_id()
                .unwrap_or("(unknown vpc)")
        );

        let check_ip = self.util.do_get("https://checkip.amazonaws.com").await?;
        let current_ip_address: Ipv4Addr = check_ip.trim().parse().map_err(|e| {
            EC2Error::new(format!(
                "Failed to convert response {} to IP Address: {e:?}",
                check_ip
            ))
        })?;

        println!("Your public IP address seems to be {current_ip_address}");
        if self.util.should_add_to_security_group() {
            match self

```

```

        .security_group_manager
        .authorize_ingress(&self.ec2, current_ip_address)
        .await
    {
        Ok(_) => println!("Security group rules updated"),
        Err(err) => eprintln!("Couldn't update security group rules:
{err:?}"),
    }
}
println!("{}", self.security_group_manager);

Ok(())
}

/// 1. Gets a list of Amazon Linux 2 AMIs from AWS Systems Manager.
Specifying the
///     '/aws/service/ami-amazon-linux-latest' path returns only the latest
AMIs.
/// 2. Gets and displays information about the available AMIs and lets you
select one.
/// 3. Gets a list of instance types that are compatible with the selected
AMI and
///     lets you select one.
/// 4. Creates an instance with the previously created key pair and security
group,
///     and the selected AMI and instance type.
/// 5. Waits for the instance to be running and then displays its
information.
pub async fn create_instance(&mut self) -> Result<(), EC2Error> {
    let ami = self.find_image().await?;

    let instance_types = self
        .ec2
        .list_instance_types(&ami.0)
        .await
        .map_err(|e| e.add_message("Could not find instance types"))?;
    println!(
        "There are several instance types that support the {} architecture of
the image.",
        ami.0
        .architecture
        .as_ref()
        .ok_or_else(|| EC2Error::new(format!("Missing architecture in
{:?}", ami.0)))?

```

```

    );
    let instance_type = self.util.select_instance_type(instance_types)?;

    println!("Creating your instance and waiting for it to start...");
    self.instance_manager
        .create(
            &self.ec2,
            ami.0
                .image_id()
                .ok_or_else(|| EC2Error::new("Could not find image ID"))?,
            instance_type,
            self.key_pair_manager.key_pair(),
            self.security_group_manager
                .security_group()
                .map(|sg| vec![sg])
                .ok_or_else(|| EC2Error::new("Could not find security
group"))?,
        )
        .await
        .map_err(|e| e.add_message("Scenario failed to create instance"))?;

    while let Err(err) = self
        .ec2
        .wait_for_instance_ready(self.instance_manager.instance_id(), None)
        .await
    {
        println!("{err}");
        if !self.util.should_continue_waiting() {
            return Err(err);
        }
    }

    println!("Your instance is ready:\n{}", self.instance_manager);

    self.display_ssh_info();

    Ok(())
}

async fn find_image(&mut self) -> Result<ScenarioImage, EC2Error> {
    let params: Vec<Parameter> = self
        .ssm
        .list_path("/aws/service/ami-amazon-linux-latest")
        .await

```

```

        .map_err(|e| e.add_message("Could not find parameters for available
images")))?
        .into_iter()
        .filter(|param| param.name().is_some_and(|name|
name.contains("amzn2")))
        .collect();
let amzn2_images: Vec<ScenarioImage> = self
    .ec2
    .list_images(params)
    .await
    .map_err(|e| e.add_message("Could not find images"))?
    .into_iter()
    .map(ScenarioImage::from)
    .collect();
println!("We will now create an instance from an Amazon Linux 2 AMI");
let ami = self.util.select_scenario_image(amzn2_images)?;
Ok(ami)
}

// 1. Stops the instance and waits for it to stop.
// 2. Starts the instance and waits for it to start.
// 3. Displays information about the instance.
// 4. Displays an SSH connection string. When an Elastic IP address is
associated
//    with the instance, the IP address stays consistent when the instance
stops
//    and starts.
pub async fn stop_and_start_instance(&self) -> Result<(), EC2Error> {
    println!("Let's stop and start your instance to see what changes.");
    println!("Stopping your instance and waiting until it's stopped...");
    self.instance_manager.stop(&self.ec2).await?;
    println!("Your instance is stopped. Restarting...");
    self.instance_manager.start(&self.ec2).await?;
    println!("Your instance is running.");
    println!("{}", self.instance_manager);
    if self.elastic_ip_manager.public_ip() == "0.0.0.0" {
        println!("Every time your instance is restarted, its public IP
address changes.");
    } else {
        println!(
            "Because you have associated an Elastic IP with your instance,
you can connect by using a consistent IP address after the instance restarts."
        );
    }
}

```

```

        self.display_ssh_info();
        Ok(())
    }

    /// 1. Allocates an Elastic IP address and associates it with the instance.
    /// 2. Displays an SSH connection string that uses the Elastic IP address.
    async fn associate_elastic_ip(&mut self) -> Result<(), EC2Error> {
        self.elastic_ip_manager.allocate(&self.ec2).await?;
        println!(
            "Allocated static Elastic IP address: {}",
            self.elastic_ip_manager.public_ip()
        );

        self.elastic_ip_manager
            .associate(&self.ec2, self.instance_manager.instance_id())
            .await?;
        println!("Associated your Elastic IP with your instance.");
        println!("You can now use SSH to connect to your instance by using the
Elastic IP.");
        self.display_ssh_info();
        Ok(())
    }

    /// Displays an SSH connection string that can be used to connect to a
    running
    /// instance.
    fn display_ssh_info(&self) {
        let ip_addr = if self.elastic_ip_manager.has_allocation() {
            self.elastic_ip_manager.public_ip()
        } else {
            self.instance_manager.instance_ip()
        };
        let key_file_path = self.key_pair_manager.key_file_path().unwrap();
        println!("To connect, open another command prompt and run the following
command:");
        println!("\nssh -i {} ec2-user@{ip_addr}\n", key_file_path.display());
        let _ = self.util.enter_to_continue();
    }

    /// 1. Disassociate and delete the previously created Elastic IP.
    /// 2. Terminate the previously created instance.
    /// 3. Delete the previously created security group.
    /// 4. Delete the previously created key pair.
    pub async fn clean_up(self) {

```

```

        println!("Let's clean everything up. This example created these
resources:");
        println!(
            "\tKey pair: {}",
            self.key_pair_manager
                .key_pair()
                .key_name()
                .unwrap_or("(unknown key pair)")
        );
        println!(
            "\tSecurity group: {}",
            self.security_group_manager.group_name()
        );
        println!(
            "\tInstance: {}",
            self.instance_manager.instance_display_name()
        );
        if self.util.should_clean_resources() {
            if let Err(err) = self.elastic_ip_manager.remove(&self.ec2).await {
                eprintln!("{err}")
            }
            if let Err(err) = self.instance_manager.delete(&self.ec2).await {
                eprintln!("{err}")
            }
            if let Err(err) = self.security_group_manager.delete(&self.ec2).await
{
                eprintln!("{err}");
            }
            if let Err(err) = self.key_pair_manager.delete(&self.ec2,
&self.util).await {
                eprintln!("{err}");
            }
        } else {
            println!("Ok, not cleaning up any resources!");
        }
    }
}

pub async fn run(mut scenario: Ec2InstanceScenario) {
    println!
    ("-----");
    println!(
        "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) get started
with instances demo."
    );
}

```

```

    );
    println!
    ("-----");

    if let Err(err) = scenario.run().await {
        eprintln!("There was an error running the scenario: {err}")
    }

    println!
    ("-----");

    scenario.clean_up().await;

    println!("Thanks for running!");
    println!
    ("-----");
}

```

EC2Impl 结构用作测试的自动模组点，其函数封装了 SDK 调用。EC2

```

use std::{net::Ipv4Addr, time::Duration};

use aws_sdk_ec2::{
    client::Waiters,
    error::ProvideErrorMetadata,
    operation::{
        allocate_address::AllocateAddressOutput,
        associate_address::AssociateAddressOutput,
    },
    types::{
        DomainType, Filter, Image, Instance, InstanceType, IpPermission, IpRange,
        KeyPairInfo,
        SecurityGroup, Tag,
    },
    Client as EC2Client,
};
use aws_sdk_ssm::types::Parameter;
use aws_smithy_runtime_api::client::waiters::error::WaiterError;

#[cfg(test)]
use mockall::automock;

```



```

#[cfg(not(test))]
pub use EC2Impl as EC2;

#[cfg(test)]
pub use MockEC2Impl as EC2;

#[derive(Clone)]
pub struct EC2Impl {
    pub client: EC2Client,
}

#[cfg_attr(test, automock)]
impl EC2Impl {
    pub fn new(client: EC2Client) -> Self {
        EC2Impl { client }
    }

    pub async fn create_key_pair(&self, name: String) -> Result<(KeyPairInfo,
String), EC2Error> {
        tracing::info!("Creating key pair {name}");
        let output = self.client.create_key_pair().key_name(name).send().await?;
        let info = KeyPairInfo::builder()
            .set_key_name(output.key_name)
            .set_key_fingerprint(output.key_fingerprint)
            .set_key_pair_id(output.key_pair_id)
            .build();
        let material = output
            .key_material
            .ok_or_else(|| EC2Error::new("Create Key Pair has no key
material"))?;
        Ok((info, material))
    }

    pub async fn list_key_pair(&self) -> Result<Vec<KeyPairInfo>, EC2Error> {
        let output = self.client.describe_key_pairs().send().await?;
        Ok(output.key_pairs.unwrap_or_default())
    }

    pub async fn delete_key_pair(&self, key_name: &str) -> Result<(), EC2Error> {
        let key_name: String = key_name.into();
        tracing::info!("Deleting key pair {key_name}");
        self.client
            .delete_key_pair()

```

```

        .key_name(key_name)
        .send()
        .await?;
    Ok(())
}

pub async fn create_security_group(
    &self,
    name: &str,
    description: &str,
) -> Result<SecurityGroup, EC2Error> {
    tracing::info!("Creating security group {name}");
    let create_output = self
        .client
        .create_security_group()
        .group_name(name)
        .description(description)
        .send()
        .await
        .map_err(EC2Error::from)?;

    let group_id = create_output
        .group_id
        .ok_or_else(|| EC2Error::new("Missing security group id after
creation"))?;

    let group = self
        .describe_security_group(&group_id)
        .await?
        .ok_or_else(|| {
            EC2Error::new(format!("Could not find security group with id
{group_id}"))
        })?;

    tracing::info!("Created security group {name} as {group_id}");

    Ok(group)
}

/// Find a single security group, by ID. Returns Err if multiple groups are
found.
pub async fn describe_security_group(
    &self,
    group_id: &str,

```

```

) -> Result<Option<SecurityGroup>, EC2Error> {
    let group_id: String = group_id.into();
    let describe_output = self
        .client
        .describe_security_groups()
        .group_ids(&group_id)
        .send()
        .await?;

    let mut groups = describe_output.security_groups.unwrap_or_default();

    match groups.len() {
        0 => Ok(None),
        1 => Ok(Some(groups.remove(0))),
        _ => Err(EC2Error::new(format!(
            "Expected single group for {group_id}"
        ))),
    }
}

/// Add an ingress rule to a security group explicitly allowing IPv4 address
/// as {ip}/32 over TCP port 22.
pub async fn authorize_security_group_ssh_ingress(
    &self,
    group_id: &str,
    ingress_ips: Vec<Ipv4Addr>,
) -> Result<(), EC2Error> {
    tracing::info!("Authorizing ingress for security group {group_id}");
    self.client
        .authorize_security_group_ingress()
        .group_id(group_id)
        .set_ip_permissions(Some(
            ingress_ips
                .into_iter()
                .map(|ip| {
                    IpPermission::builder()
                        .ip_protocol("tcp")
                        .from_port(22)
                        .to_port(22)
                        .ip_ranges(IpRange::builder().cidr_ip(format!(
                            "{ip}/32"))).build())
                        .build()
                })
        ))
        .collect(),

```

```

        ))
        .send()
        .await?;
    Ok(())
}

pub async fn delete_security_group(&self, group_id: &str) -> Result<(),
EC2Error> {
    tracing::info!("Deleting security group {group_id}");
    self.client
        .delete_security_group()
        .group_id(group_id)
        .send()
        .await?;
    Ok(())
}

pub async fn list_images(&self, ids: Vec<Parameter>) -> Result<Vec<Image>,
EC2Error> {
    let image_ids = ids.into_iter().filter_map(|p| p.value).collect();
    let output = self
        .client
        .describe_images()
        .set_image_ids(Some(image_ids))
        .send()
        .await?;

    let images = output.images.unwrap_or_default();
    if images.is_empty() {
        Err(EC2Error::new("No images for selected AMIs"))
    } else {
        Ok(images)
    }
}

/// List instance types that match an image's architecture and are free tier
eligible.
pub async fn list_instance_types(&self, image: &Image) ->
Result<Vec<InstanceType>, EC2Error> {
    let architecture = format!(
        "{}",
        image.architecture().ok_or_else(|| EC2Error::new(format!(
            "Image {:?} does not have a listed architecture",
            image.image_id()

```

```

    )))?
  );
  let free_tier_eligible_filter = Filter::builder()
    .name("free-tier-eligible")
    .values("false")
    .build();
  let supported_architecture_filter = Filter::builder()
    .name("processor-info.supported-architecture")
    .values(architecture)
    .build();
  let response = self
    .client
    .describe_instance_types()
    .filters(free_tier_eligible_filter)
    .filters(supported_architecture_filter)
    .send()
    .await?;

  Ok(response
    .instance_types
    .unwrap_or_default()
    .into_iter()
    .filter_map(|iti| iti.instance_type)
    .collect())
}

pub async fn create_instance<'a>(
  &self,
  image_id: &'a str,
  instance_type: InstanceType,
  key_pair: &'a KeyPairInfo,
  security_groups: Vec<&'a SecurityGroup>,
) -> Result<String, EC2Error> {
  let run_instances = self
    .client
    .run_instances()
    .image_id(image_id)
    .instance_type(instance_type)
    .key_name(
      key_pair
        .key_name()
        .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?),
    )

```

```

        .set_security_group_ids(Some(
            security_groups
                .iter()
                .filter_map(|sg| sg.group_id.clone())
                .collect(),
        ))
        .min_count(1)
        .max_count(1)
        .send()
        .await?;

if run_instances.instances().is_empty() {
    return Err(EC2Error::new("Failed to create instance"));
}

let instance_id = run_instances.instances()[0].instance_id().unwrap();
let response = self
    .client
    .create_tags()
    .resources(instance_id)
    .tags(
        Tag::builder()
            .key("Name")
            .value("From SDK Examples")
            .build(),
    )
    .send()
    .await;

match response {
    Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
    Err(err) => {
        tracing::info!("Error applying tags to {instance_id}: {err:?}");
        return Err(err.into());
    }
}

tracing::info!("Instance is created.");

Ok(instance_id.to_string())
}

/// Wait for an instance to be ready and status ok (default wait 60 seconds)
pub async fn wait_for_instance_ready(

```

```

        &self,
        instance_id: &str,
        duration: Option<Duration>,
    ) -> Result<(), EC2Error> {
        self.client
            .wait_until_instance_status_ok()
            .instance_ids(instance_id)
            .wait(duration.unwrap_or(Duration::from_secs(60)))
            .await
            .map_err(|err| match err {
                WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                    "Exceeded max time ({}s) waiting for instance to start.",
                    exceeded.max_wait().as_secs()
                )),
                _ => EC2Error::from(err),
            })?;
        Ok(())
    }

    pub async fn describe_instance(&self, instance_id: &str) -> Result<Instance,
    EC2Error> {
        let response = self
            .client
            .describe_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        let instance = response
            .reservations()
            .first()
            .ok_or_else(|| EC2Error::new(format!("No instance reservations for
{instance_id}")))?
            .instances()
            .first()
            .ok_or_else(|| {
                EC2Error::new(format!("No instances in reservation for
{instance_id}"))
            })?;

        Ok(instance.clone())
    }

```

```
pub async fn start_instance(&self, instance_id: &str) -> Result<(), EC2Error>
{
    tracing::info!("Starting instance {instance_id}");

    self.client
        .start_instances()
        .instance_ids(instance_id)
        .send()
        .await?;

    tracing::info!("Started instance.");

    Ok(())
}

pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error>
{
    tracing::info!("Stopping instance {instance_id}");

    self.client
        .stop_instances()
        .instance_ids(instance_id)
        .send()
        .await?;

    self.wait_for_instance_stopped(instance_id, None).await?;

    tracing::info!("Stopped instance.");

    Ok(())
}

pub async fn reboot_instance(&self, instance_id: &str) -> Result<(),
EC2Error> {
    tracing::info!("Rebooting instance {instance_id}");

    self.client
        .reboot_instances()
        .instance_ids(instance_id)
        .send()
        .await?;

    Ok(())
}
```



```

pub async fn wait_for_instance_stopped(
    &self,
    instance_id: &str,
    duration: Option<Duration>,
) -> Result<(), EC2Error> {
    self.client
        .wait_until_instance_stopped()
        .instance_ids(instance_id)
        .wait(duration.unwrap_or(Duration::from_secs(60)))
        .await
        .map_err(|err| match err {
            WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                "Exceeded max time ({}s) waiting for instance to stop.",
                exceeded.max_wait().as_secs(),
            )),
            _ => EC2Error::from(err),
        })?;
    Ok(())
}

pub async fn delete_instance(&self, instance_id: &str) -> Result<(),
EC2Error> {
    tracing::info!("Deleting instance with id {instance_id}");
    self.stop_instance(instance_id).await?;
    self.client
        .terminate_instances()
        .instance_ids(instance_id)
        .send()
        .await?;
    self.wait_for_instance_terminated(instance_id).await?;
    tracing::info!("Terminated instance with id {instance_id}");
    Ok(())
}

async fn wait_for_instance_terminated(&self, instance_id: &str) -> Result<(),
EC2Error> {
    self.client
        .wait_until_instance_terminated()
        .instance_ids(instance_id)
        .wait(Duration::from_secs(60))
        .await
        .map_err(|err| match err {
            WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(

```

```

        "Exceeded max time ({})s waiting for instance to terminate.",
        exceeded.max_wait().as_secs(),
    )),
    _ => EC2Error::from(err),
  })?;
Ok(())
}

pub async fn allocate_ip_address(&self) -> Result<AllocateAddressOutput,
EC2Error> {
    self.client
        .allocate_address()
        .domain(DomainType::Vpc)
        .send()
        .await
        .map_err(EC2Error::from)
}

pub async fn deallocate_ip_address(&self, allocation_id: &str) -> Result<(),
EC2Error> {
    self.client
        .release_address()
        .allocation_id(allocation_id)
        .send()
        .await?;
    Ok(())
}

pub async fn associate_ip_address(
    &self,
    allocation_id: &str,
    instance_id: &str,
) -> Result<AssociateAddressOutput, EC2Error> {
    let response = self
        .client
        .associate_address()
        .allocation_id(allocation_id)
        .instance_id(instance_id)
        .send()
        .await?;
    Ok(response)
}

```

```

    pub async fn disassociate_ip_address(&self, association_id: &str) ->
Result<(), EC2Error> {
    self.client
        .disassociate_address()
        .association_id(association_id)
        .send()
        .await?;
    Ok(())
}
}

#[derive(Debug)]
pub struct EC2Error(String);
impl EC2Error {
    pub fn new(value: impl Into<String>) -> Self {
        EC2Error(value.into())
    }

    pub fn add_message(self, message: impl Into<String>) -> Self {
        EC2Error(format!("{}", message.into(), self.0))
    }
}

impl<T: ProvideErrorMetadata> From<T> for EC2Error {
    fn from(value: T) -> Self {
        EC2Error(format!(
            "{}: {}",
            value
                .code()
                .map(String::from)
                .unwrap_or("unknown code".into()),
            value
                .message()
                .map(String::from)
                .unwrap_or("missing reason".into()),
        ))
    }
}

impl std::error::Error for EC2Error {}

impl std::fmt::Display for EC2Error {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}

```

```
    }
}
```

SSM 结构体用作测试的自动模拟点，其函数封装了 SSM SDK 调用。

```
use aws_sdk_ssm::{types::Parameter, Client};
use aws_smithy_async::future::pagination_stream::TryFlatMap;

use crate::ec2::EC2Error;

#[cfg(test)]
use mockall::automock;

#[cfg(not(test))]
pub use SSMImpl as SSM;

#[cfg(test)]
pub use MockSSMImpl as SSM;

pub struct SSMImpl {
    inner: Client,
}

#[cfg_attr(test, automock)]
impl SSMImpl {
    pub fn new(inner: Client) -> Self {
        SSMImpl { inner }
    }

    pub async fn list_path(&self, path: &str) -> Result<Vec<Parameter>, EC2Error>
    {
        let maybe_params: Vec<Result<Parameter, _>> = TryFlatMap::new(
            self.inner
                .get_parameters_by_path()
                .path(path)
                .into_paginator()
                .send(),
        )
        .flat_map(|item| item.parameters.unwrap_or_default())
        .collect()
        .await;
    }
}
```

```

        // Fail on the first error
        let params = maybe_params
            .into_iter()
            .collect::<Result<Vec<Parameter>, _>>()?;
        Ok(params)
    }
}

```

此场景使用多个“管理器”风格的结构体，来处理对在整个场景中创建和删除的资源访问。

```

use aws_sdk_ec2::operation::{
    allocate_address::AllocateAddressOutput,
    associate_address::AssociateAddressOutput,
};

use crate::ec2::{EC2Error, EC2};

/// ElasticIpManager tracks the lifecycle of a public IP address, including its
/// allocation from the global pool and association with a specific instance.
#[derive(Debug, Default)]
pub struct ElasticIpManager {
    elastic_ip: Option<AllocateAddressOutput>,
    association: Option<AssociateAddressOutput>,
}

impl ElasticIpManager {
    pub fn has_allocation(&self) -> bool {
        self.elastic_ip.is_some()
    }

    pub fn public_ip(&self) -> &str {
        if let Some(allocation) = &self.elastic_ip {
            if let Some(addr) = allocation.public_ip() {
                return addr;
            }
        }
        "0.0.0.0"
    }

    pub async fn allocate(&mut self, ec2: &EC2) -> Result<(), EC2Error> {
        let allocation = ec2.allocate_ip_address().await?;
    }
}

```

```

        self.elastic_ip = Some(allocation);
        Ok(())
    }

    pub async fn associate(&mut self, ec2: &EC2, instance_id: &str) -> Result<(),
    EC2Error> {
        if let Some(allocation) = &self.elastic_ip {
            if let Some(allocation_id) = allocation.allocation_id() {
                let association = ec2.associate_ip_address(allocation_id,
                instance_id).await?;
                self.association = Some(association);
                return Ok(());
            }
        }
        Err(EC2Error::new("No ip address allocation to associate"))
    }

    pub async fn remove(mut self, ec2: &EC2) -> Result<(), EC2Error> {
        if let Some(association) = &self.association {
            if let Some(association_id) = association.association_id() {
                ec2.disassociate_ip_address(association_id).await?;
            }
        }
        self.association = None;
        if let Some(allocation) = &self.elastic_ip {
            if let Some(allocation_id) = allocation.allocation_id() {
                ec2.deallocate_ip_address(allocation_id).await?;
            }
        }
        self.elastic_ip = None;
        Ok(())
    }
}

use std::fmt::Display;

use aws_sdk_ec2::types::{Instance, InstanceType, KeyPairInfo, SecurityGroup};

use crate::ec2::{EC2Error, EC2};

/// InstanceManager wraps the lifecycle of an EC2 Instance.
#[derive(Debug, Default)]
pub struct InstanceManager {

```

```

        instance: Option<Instance>,
    }

impl InstanceManager {
    pub fn instance_id(&self) -> &str {
        if let Some(instance) = &self.instance {
            if let Some(id) = instance.instance_id() {
                return id;
            }
        }
        "Unknown"
    }

    pub fn instance_name(&self) -> &str {
        if let Some(instance) = &self.instance {
            if let Some(tag) = instance.tags().iter().find(|e| e.key() ==
Some("Name")) {
                if let Some(value) = tag.value() {
                    return value;
                }
            }
        }
        "Unknown"
    }

    pub fn instance_ip(&self) -> &str {
        if let Some(instance) = &self.instance {
            if let Some(public_ip_address) = instance.public_ip_address() {
                return public_ip_address;
            }
        }
        "0.0.0.0"
    }

    pub fn instance_display_name(&self) -> String {
        format!("{}", self.instance_name(), self.instance_id())
    }

    /// Create an EC2 instance with the given ID on a given type, using a
    /// generated KeyPair and applying a list of security groups.
    pub async fn create(
        &mut self,
        ec2: &EC2,
        image_id: &str,

```

```

        instance_type: InstanceType,
        key_pair: &KeyPairInfo,
        security_groups: Vec<&SecurityGroup>,
    ) -> Result<(), EC2Error> {
        let instance_id = ec2
            .create_instance(image_id, instance_type, key_pair, security_groups)
            .await?;
        let instance = ec2.describe_instance(&instance_id).await?;
        self.instance = Some(instance);
        Ok(())
    }

    /// Start the managed EC2 instance, if present.
    pub async fn start(&self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.start_instance(self.instance_id()).await?;
        }
        Ok(())
    }

    /// Stop the managed EC2 instance, if present.
    pub async fn stop(&self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.stop_instance(self.instance_id()).await?;
        }
        Ok(())
    }

    pub async fn reboot(&self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.reboot_instance(self.instance_id()).await?;
            ec2.wait_for_instance_stopped(self.instance_id(), None)
                .await?;
            ec2.wait_for_instance_ready(self.instance_id(), None)
                .await?;
        }
        Ok(())
    }

    /// Terminate and delete the managed EC2 instance, if present.
    pub async fn delete(self, ec2: &EC2) -> Result<(), EC2Error> {
        if self.instance.is_some() {
            ec2.delete_instance(self.instance_id()).await?;
        }
    }

```



```

        Ok(())
    }
}

impl Display for InstanceManager {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        if let Some(instance) = &self.instance {
            writeln!(f, "\tID: {}",
instance.instance_id().unwrap_or("(Unknown)"));
            writeln!(
                f,
                "\tImage ID: {}",
                instance.image_id().unwrap_or("(Unknown)")
            )?;
            writeln!(
                f,
                "\tInstance type: {}",
                instance
                    .instance_type()
                    .map(|it| format!("{it}"))
                    .unwrap_or("(Unknown)".to_string())
            )?;
            writeln!(
                f,
                "\tKey name: {}",
                instance.key_name().unwrap_or("(Unknown)")
            )?;
            writeln!(f, "\tVPC ID: {}",
instance.vpc_id().unwrap_or("(Unknown)"));
            writeln!(
                f,
                "\tPublic IP: {}",
                instance.public_ip_address().unwrap_or("(Unknown)")
            )?;
            let instance_state = instance
                .state
                .as_ref()
                .map(|is| {
                    is.name()
                        .map(|isn| format!("{isn}"))
                        .unwrap_or("(Unknown)".to_string())
                })
                .unwrap_or("(Unknown)".to_string());
            writeln!(f, "\tState: {instance_state}");
        }
    }
}

```

```

        } else {
            writeln!(f, "\tNo loaded instance"?);
        }
        Ok(())
    }
}

use std::{env, path::PathBuf};

use aws_sdk_ec2::types::KeyPairInfo;

use crate::ec2::{EC2Error, EC2};

use super::util::Util;

/// KeyPairManager tracks a KeyPairInfo and the path the private key has been
/// written to, if it's been created.
#[derive(Debug)]
pub struct KeyPairManager {
    key_pair: KeyPairInfo,
    key_file_path: Option<PathBuf>,
    key_file_dir: PathBuf,
}

impl KeyPairManager {
    pub fn new() -> Self {
        Self::default()
    }

    pub fn key_pair(&self) -> &KeyPairInfo {
        &self.key_pair
    }

    pub fn key_file_path(&self) -> Option<&PathBuf> {
        self.key_file_path.as_ref()
    }

    pub fn key_file_dir(&self) -> &PathBuf {
        &self.key_file_dir
    }

    /// Creates a key pair that can be used to securely connect to an EC2
    instance.

```

```

    /// The returned key pair contains private key information that cannot be
    retrieved
    /// again. The private key data is stored as a .pem file.
    ///
    /// :param key_name: The name of the key pair to create.
    pub async fn create(
        &mut self,
        ec2: &EC2,
        util: &Util,
        key_name: String,
    ) -> Result<KeyPairInfo, EC2Error> {
        let (key_pair, material) =
ec2.create_key_pair(key_name.clone()).await.map_err(|e| {
            self.key_pair =
KeyPairInfo::builder().key_name(key_name.clone()).build();
            e.add_message(format!("Couldn't create key {key_name}"))
        })?;

        let path = self.key_file_dir.join(format!("{key_name}.pem"));

        // Save the key_pair information immediately, so it can get cleaned up if
        write_secure fails.
        self.key_file_path = Some(path.clone());
        self.key_pair = key_pair.clone();

        util.write_secure(&key_name, &path, material)?;

        Ok(key_pair)
    }

    pub async fn delete(self, ec2: &EC2, util: &Util) -> Result<(), EC2Error> {
        if let Some(key_name) = self.key_pair.key_name() {
            ec2.delete_key_pair(key_name).await?;
            if let Some(key_path) = self.key_file_path() {
                if let Err(err) = util.remove(key_path) {
                    eprintln!("Failed to remove {key_path:?} ({err:?})");
                }
            }
        }
        Ok(())
    }

    pub async fn list(&self, ec2: &EC2) -> Result<Vec<KeyPairInfo>, EC2Error> {
        ec2.list_key_pair().await
    }

```

```

    }
}

impl Default for KeyPairManager {
    fn default() -> Self {
        KeyPairManager {
            key_pair: KeyPairInfo::builder().build(),
            key_file_path: Default::default(),
            key_file_dir: env::temp_dir(),
        }
    }
}

use std::net::Ipv4Addr;

use aws_sdk_ec2::types::SecurityGroup;

use crate::ec2::{EC2Error, EC2};

/// SecurityGroupManager tracks the lifecycle of a SecurityGroup for an instance,
/// including adding a rule to allow SSH from a public IP address.
#[derive(Debug, Default)]
pub struct SecurityGroupManager {
    group_name: String,
    group_description: String,
    security_group: Option<SecurityGroup>,
}

impl SecurityGroupManager {
    pub async fn create(
        &mut self,
        ec2: &EC2,
        group_name: &str,
        group_description: &str,
    ) -> Result<(), EC2Error> {
        self.group_name = group_name.into();
        self.group_description = group_description.into();

        self.security_group = Some(
            ec2.create_security_group(group_name, group_description)
                .await
                .map_err(|e| e.add_message("Couldn't create security group"))?,
        );
    }
}

```

```

        Ok(())
    }

    pub async fn authorize_ingress(&self, ec2: &EC2, ip_address: Ipv4Addr) ->
Result<(), EC2Error> {
        if let Some(sg) = &self.security_group {
            ec2.authorize_security_group_ssh_ingress(
                sg.group_id()
                    .ok_or_else(|| EC2Error::new("Missing security group ID"))?,
                vec![ip_address],
            )
                .await?;
        };

        Ok(())
    }

    pub async fn delete(self, ec2: &EC2) -> Result<(), EC2Error> {
        if let Some(sg) = &self.security_group {
            ec2.delete_security_group(
                sg.group_id()
                    .ok_or_else(|| EC2Error::new("Missing security group ID"))?,
            )
                .await?;
        };

        Ok(())
    }

    pub fn group_name(&self) -> &str {
        &self.group_name
    }

    pub fn vpc_id(&self) -> Option<&str> {
        self.security_group.as_ref().and_then(|sg| sg.vpc_id())
    }

    pub fn security_group(&self) -> Option<&SecurityGroup> {
        self.security_group.as_ref()
    }
}

impl std::fmt::Display for SecurityGroupManager {

```

```

fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
    match &self.security_group {
        Some(sg) => {
            writeln!(
                f,
                "Security group: {}",
                sg.group_name().unwrap_or("(unknown group)")
            )?;
            writeln!(f, "\tID: {}", sg.group_id().unwrap_or("(unknown group
id)"))?;
            writeln!(f, "\tVPC: {}", sg.vpc_id().unwrap_or("(unknown group
vpc)"))?;
            if !sg.ip_permissions().is_empty() {
                writeln!(f, "\tInbound Permissions:");
                for permission in sg.ip_permissions() {
                    writeln!(f, "\t\t{permission:?}")?;
                }
            }
            Ok(())
        }
        None => writeln!(f, "No security group loaded."),
    }
}
}

```

场景的主入口点。

```

use ec2_code_examples::{
    ec2::EC2,
    getting_started::{
        scenario::{run, Ec2InstanceScenario},
        util::UtilImpl,
    },
    ssm::SSM,
};

#[tokio::main]
async fn main() {
    tracing_subscriber::fmt::init();
    let sdk_config = aws_config::load_from_env().await;
    let ec2 = EC2::new(aws_sdk_ec2::Client::new(&sdk_config));

```

```
let ssm = SSM::new(aws_sdk_ssm::Client::new(&sdk_config));  
let util = UtilImpl {};  
let scenario = Ec2InstanceScenario::new(ec2, ssm, util);  
run(scenario).await;  
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Rust API Reference》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](#)

Swift

适用于 Swift 的 SDK

Note

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

Package.swift 文件。

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "ec2-scenario",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/aws-labs/aws-sdk-swift",
            from: "1.4.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
        // or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "ec2-scenario",
```



```

        dependencies: [
            .product(name: "AWSEC2", package: "aws-sdk-swift"),
            .product(name: "AWSSSM", package: "aws-sdk-swift"),
            .product(name: "ArgumentParser", package: "swift-argument-
parser")
        ],
        path: "Sources")
    ]
)

```

entry.swift 文件。

```

// An example that shows how to use the AWS SDK for Swift to perform a variety
// of operations using Amazon Elastic Compute Cloud (EC2).
//

import ArgumentParser
import Foundation
import AWSEC2

// Allow waiters to be used.

import class SmithyWaitersAPI.Waiter
import struct SmithyWaitersAPI.WaiterOptions

import AWSSSM

struct ExampleCommand: ParsableCommand {
    @Option(help: "The AWS Region to run AWS API calls in.")
    var awsRegion = "us-east-1"

    @Option(
        help: ArgumentHelp("The level of logging for the Swift SDK to perform."),
        completion: .list([
            "critical",
            "debug",
            "error",
            "info",
            "notice",
            "trace",
            "warning"
        ])
    )
}

```

```

    ])
  )
  var logLevel: String = "error"

  static var configuration = CommandConfiguration(
    commandName: "ec2-scenario",
    abstract: ""
    Performs various operations to demonstrate the use of Amazon EC2 using
the
    AWS SDK for Swift.
    "",
    discussion: ""
    ""
  )

  /// Called by ``main()`` to run the bulk of the example.
  func runAsync() async throws {
    let ssmConfig = try await SSMClient.SSMClientConfiguration(region:
awsRegion)
    let ssmClient = SSMClient(config: ssmConfig)

    let ec2Config = try await EC2Client.EC2ClientConfiguration(region:
awsRegion)
    let ec2Client = EC2Client(config: ec2Config)

    let example = Example(ec2Client: ec2Client, ssmClient: ssmClient)

    await example.run()
  }
}

class Example {
  let ec2Client: EC2Client
  let ssmClient: SSMClient

  // Storage for AWS EC2 properties.

  var keyName: String? = nil
  var securityGroupId: String? = nil
  var instanceId: String? = nil
  var allocationId: String? = nil
  var associationId: String? = nil

  init(ec2Client: EC2Client, ssmClient: SSMClient) {

```

```

        self.ec2Client = ec2Client
        self.ssmClient = ssmClient
    }

    /// The example's main body.
    func run() async {
        //=====
        // 1. Create an RSA key pair, saving the private key as a `.pem` file.
        //    Create a `defer` block that will delete the private key when the
        //    program exits.
        //=====

        print("Creating an RSA key pair...")

        keyName = self.tempName(prefix: "ExampleKeyName")
        let keyUrl = await self.createKeyPair(name: keyName!)

        guard let keyUrl else {
            print("*** Failed to create the key pair!")
            return
        }

        print("Created the private key at: \(keyUrl.absoluteString)")

        // Schedule deleting the private key file to occur automatically when
        // the program exits, no matter how it exits.

        defer {
            do {
                try FileManager.default.removeItem(at: keyUrl)
            } catch {
                print("*** Failed to delete the private key at
                \(keyUrl.absoluteString)")
            }
        }

        //=====
        // 2. List the key pairs by calling `DescribeKeyPairs`.
        //=====

        print("Describing available key pairs...")
        await self.describeKeyPairs()

        //=====
    }

```

```

// 3. Create a security group for the default VPC, and add an inbound
//    rule to allow SSH from the current computer's public IPv4
//    address.
//=====

print("Creating the security group...")

let secGroupName = self.tempName(prefix: "ExampleSecurityGroup")
let ipAddress = self.getMyIPAddress()

guard let ipAddress else {
    print("*** Unable to get the device's IP address.")
    return
}

print("IP address is: \(ipAddress)")

securityGroupId = await self.createSecurityGroup(
    name: secGroupName,
    description: "An example security group created using the AWS SDK for
Swift"
)

if securityGroupId == nil {
    await cleanUp()
    return
}

print("Created security group: \(securityGroupId ?? "<unknown>")")

if !(await self.authorizeSecurityGroupIngress(groupId: securityGroupId!,
ipAddress: ipAddress)) {
    await cleanUp()
    return
}

//=====
// 4. Display security group information for the new security group
//    using DescribeSecurityGroups.
//=====

if !(await self.describeSecurityGroups(groupId: securityGroupId!)) {
    await cleanUp()
    return
}

```

```

}

//=====
// 5. Get a list of Amazon Linux 2023 AMIs and pick one (SSM is the
//    best practice), using path and then filter the list after the
//    fact to include "al2023" in the Name field
//    (ssm.GetParametersByPath). Paginate to get all images.
//=====

print("Searching available images for Amazon Linux 2023 images...")

let options = await self.findAMIsMatchingFilter("al2023")

//=====
// 6. The information in the AMI options isn't great, so make a list
//    of the image IDs (the "Value" field in the AMI options) and get
//    more information about them from EC2. Display the Description
//    field and select one of them (DescribeImages with ImageIds
//    filter).
//=====

print("Images matching Amazon Linux 2023:")

var imageIds: [String] = []
for option in options {
    guard let id = option.value else {
        continue
    }
    imageIds.append(id)
}

let images = await self.describeImages(imageIds)

// This is where you would normally let the user choose which AMI to
// use. However, for this example, we're just going to use the first
// one, whatever it is.

let chosenImage = images[0]

//=====
// 7. Get a list of instance types that are compatible with the
//    selected AMI's architecture (such as "x86_64") and are either
//    small or micro. Select one (DescribeInstanceTypes).
//=====

```

```
print("Getting the instance types compatible with the selected image...")

guard let arch = chosenImage.architecture else {
    print("*** The selected image doesn't have a valid architecture.")
    await cleanUp()
    return
}

let imageTypes = await self.getMatchingInstanceTypes(architecture: arch)

for type in imageTypes {
    guard let instanceType = type.instanceType else {
        continue
    }
    print("    \(instanceType.rawValue)")
}

// This example selects the first returned instance type. A real-world
// application would probably ask the user to select one here.

let chosenInstanceType = imageTypes[0]

//=====
// 8. Create an instance with the key pair, security group, AMI, and
//    instance type (RunInstances).
//=====

print("Creating an instance...")

guard let imageId = chosenImage.imageId else {
    print("*** Cannot start image without a valid image ID.")
    await cleanUp()
    return
}

guard let instanceType = chosenInstanceType.instanceType else {
    print("*** Unable to start image without a valid image type.")
    await cleanUp()
    return
}

let instance = await self.runInstance(
    imageId: imageId,
    instanceType: instanceType,
```

```

        keyPairName: keyName!,
        securityGroups: [securityGroupId!]
    )

    guard let instance else {
        await cleanUp()
        return
    }

    instanceId = instance.instanceId
    if instanceId == nil {
        print("*** Instance is missing an ID. Canceling.")
        await cleanUp()
        return
    }

    //=====
    // 9. Wait for the instance to be ready and then display its
    //    information (DescribeInstances).
    //=====

    print("Waiting a few seconds to let the instance come up...")

    do {
        try await Task.sleep(for: .seconds(20))
    } catch {
        print("*** Error pausing the task.")
    }
    print("Success! Your new instance is ready:")

    //=====
    // 10. Display SSH connection info for the instance.
    //=====

    var runningInstance = await self.describeInstance(instanceId:
instanceId!)

    if (runningInstance != nil) && (runningInstance!.publicIpAddress != nil)
{
        print("\nYou can SSH to this instance using the following command:")
        print("ssh -i \(keyUrl.path) ec2-user@
\(runningInstance!.publicIpAddress!)")
    }

```

```

//=====
// 11. Stop the instance and wait for it to stop (StopInstances).
//=====

print("Stopping the instance...")

if !(await self.stopInstance(instanceId: instanceId!, waitUntilStopped:
true)) {
    await cleanUp()
    return
}

//=====
// 12. Start the instance and wait for it to start (StartInstances).
//=====

print("Starting the instance again...")

if !(await self.startInstance(instanceId: instanceId!, waitUntilStarted:
true)) {
    await cleanUp()
    return
}

//=====
// 13. Display SSH connection info for the instance. Note that it's
//      changed.
//=====

runningInstance = await self.describeInstance(instanceId: instanceId!)
if (runningInstance != nil) && (runningInstance!.publicIpAddress != nil)
{
    print("\nYou can SSH to this instance using the following command.")
    print("This is probably different from when the instance was running
before.")
    print("ssh -i \$(keyUrl.path) ec2-user@
\$(runningInstance!.publicIpAddress!)")
}

//=====
// 14. Allocate an elastic IP and associate it with the instance
//      (AllocateAddress and AssociateAddress).
//=====

```



```

        allocationId = await self.allocateAddress()

        if allocationId == nil {
            await cleanUp()
            return
        }

        associationId = await self.associateAddress(instanceId: instanceId!,
allocationId: allocationId)

        if associationId == nil {
            await cleanUp()
            return
        }

        //=====
        // 15. Display SSH connection info for the connection. Note that the
        //      public IP is now the Elastic IP, which stays constant.
        //=====

        runningInstance = await self.describeInstance(instanceId: instanceId!)
        if (runningInstance != nil) && (runningInstance!.publicIpAddress != nil)
{
            print("\nYou can SSH to this instance using the following command.")
            print("This has changed again, and is now the Elastic IP.")
            print("ssh -i \$(keyUrl.path) ec2-user@
\$(runningInstance!.publicIpAddress!)")
        }

        //=====
        // Handle all cleanup tasks
        //=====

        await cleanUp()
    }

    /// Clean up by discarding and closing down all allocated EC2 items:
    ///
    /// * Elastic IP allocation and association
    /// * Terminate the instance
    /// * Delete the security group
    /// * Delete the key pair
    func cleanUp() async {
        //=====

```

```

// 16. Disassociate and delete the Elastic IP (DisassociateAddress and
//      ReleaseAddress).
//=====

if associationId != nil {
    await self.disassociateAddress(associationId: associationId!)
}

if allocationId != nil {
    await self.releaseAddress(allocationId: allocationId!)
}

//=====
// 17. Terminate the instance and wait for it to terminate
//      (TerminateInstances).
//=====

if instanceId != nil {
    print("Terminating the instance...")
    _ = await self.terminateInstance(instanceId: instanceId!,
waitUntilTerminated: true)
}

//=====
// 18. Delete the security group (DeleteSecurityGroup).
//=====

if securityGroupId != nil {
    print("Deleting the security group...")
    _ = await self.deleteSecurityGroup(groupId: securityGroupId!)
}

//=====
// 19. Delete the key pair (DeleteKeyPair).
//=====

if keyName != nil {
    print("Deleting the key pair...")
    _ = await self.deleteKeyPair(keyPair: keyName!)
}
}

/// Create a new RSA key pair and save the private key to a randomly-named
/// file in the temporary directory.

```

```
///
/// - Parameter name: The name of the key pair to create.
///
/// - Returns: The URL of the newly created `.pem` file or `nil` if unable
///   to create the key pair.
func createKeyPair(name: String) async -> URL? {
    do {
        let output = try await ec2Client.createKeyPair(
            input: CreateKeyPairInput(
                keyName: name
            )
        )

        guard let keyMaterial = output.keyMaterial else {
            return nil
        }

        // Build the URL of the temporary private key file.

        let fileURL = URL.temporaryDirectory
            .appendingPathComponent(name)
            .appendingPathExtension("pem")

        do {
            try keyMaterial.write(to: fileURL, atomically: true, encoding:
String.Encoding.utf8)
            return fileURL
        } catch {
            print("*** Failed to write the private key.")
            return nil
        }
    } catch {
        print("*** Unable to create the key pair.")
        return nil
    }
}

/// Describe the key pairs associated with the user by outputting each key
/// pair's name and fingerprint.
func describeKeyPairs() async {
    do {
        let output = try await ec2Client.describeKeyPairs(
            input: DescribeKeyPairsInput()
        )
    }
}
```

```

        guard let keyPairs = output.keyPairs else {
            print("*** No key pairs list available.")
            return
        }

        for keyPair in keyPairs {
            print(keyPair.keyName ?? "<unknown>", ":",
keyPair.keyFingerprint ?? "<unknown>")
        }
    } catch {
        print("*** Error: Unable to obtain a key pair list.")
    }
}

/// Delete an EC2 key pair.
///
/// - Parameter keyPair: The name of the key pair to delete.
///
/// - Returns: `true` if the key pair is deleted successfully; otherwise
///   `false`.
func deleteKeyPair(keyPair: String) async -> Bool {
    do {
        _ = try await ec2Client.deleteKeyPair(
            input: DeleteKeyPairInput(
                keyName: keyPair
            )
        )

        return true
    } catch {
        print("*** Error deleting the key pair:
\\(error.localizedDescription)")
        return false
    }
}

/// Return a list of AMI names that contain the specified string.
///
/// - Parameter filter: A string that must be contained in all returned
///   AMI names.
///
/// - Returns: An array of the parameters matching the specified substring.

```

```

func findAMIsMatchingFilter(_ filter: String) async ->
[SSMClientTypes.Parameter] {
    var parameterList: [SSMClientTypes.Parameter] = []
    var matchingAMIs: [SSMClientTypes.Parameter] = []

    do {
        let pages = ssmClient.getParametersByPathPaginated(
            input: GetParametersByPathInput(
                path: "/aws/service/ami-amazon-linux-latest"
            )
        )

        for try await page in pages {
            guard let parameters = page.parameters else {
                return matchingAMIs
            }

            for parameter in parameters {
                parameterList.append(parameter)
            }
        }

        print("Found \(parameterList.count) images total:")
        for parameter in parameterList {
            guard let name = parameter.name else {
                continue
            }
            print("    \(name)")

            if name.contains(filter) {
                matchingAMIs.append(parameter)
            }
        }
    } catch {
        return matchingAMIs
    }

    return matchingAMIs
}

/// Return a list of instance types matching the specified architecture
/// and instance sizes.
///
/// - Parameters:

```

```

/// - architecture: The architecture of the instance types to return, as
///   a member of `EC2ClientTypes.ArchitectureValues`.
/// - sizes: An array of one or more strings identifying sizes of
///   instance type to accept.
///
/// - Returns: An array of `EC2ClientTypes.InstanceTypeInfo` records
///   describing the instance types matching the given requirements.
func getMatchingInstanceTypes(architecture: EC2ClientTypes.ArchitectureValues
= EC2ClientTypes.ArchitectureValues.x8664,
                           sizes: [String] = ["*.micro", "/*.small"]) async
    -> [EC2ClientTypes.InstanceTypeInfo] {
    var instanceTypes: [EC2ClientTypes.InstanceTypeInfo] = []

    let archFilter = EC2ClientTypes.Filter(
        name: "processor-info.supported-architecture",
        values: [architecture.rawValue]
    )
    let sizeFilter = EC2ClientTypes.Filter(
        name: "instance-type",
        values: sizes
    )

    do {
        let pages = ec2Client.describeInstanceTypesPaginated(
            input: DescribeInstanceTypesInput(
                filters: [archFilter, sizeFilter]
            )
        )

        for try await page in pages {
            guard let types = page.instanceTypes else {
                return []
            }

            instanceTypes += types
        }
    } catch {
        print("*** Error getting image types: \(error.localizedDescription)")
        return []
    }

    return instanceTypes
}

```

```
/// Get the latest information about the specified instance and output it
/// to the screen, returning the instance details to the caller.
///
/// - Parameters:
///   - instanceId: The ID of the instance to provide details about.
///   - stateFilter: The state to require the instance to be in.
///
/// - Returns: The instance's details as an `EC2ClientTypes.Instance` object.
func describeInstance(instanceId: String,
                      stateFilter: EC2ClientTypes.InstanceStateName? =
EC2ClientTypes.InstanceStateName.running) async
    -> EC2ClientTypes.Instance? {
    do {
        let pages = ec2Client.describeInstancesPaginated(
            input: DescribeInstancesInput(
                instanceIds: [instanceId]
            )
        )

        for try await page in pages {
            guard let reservations = page.reservations else {
                continue
            }

            for reservation in reservations {
                guard let instances = reservation.instances else {
                    continue
                }

                for instance in instances {
                    guard let state = instance.state else {
                        print("*** Instance is missing its state...")
                        continue
                    }
                    let instanceState = state.name

                    if stateFilter != nil && (instanceState != stateFilter) {
                        continue
                    }

                    let instanceTypeName: String
                    if instance.instanceType == nil {
                        instanceTypeName = "<N/A>"
                    } else {
```

```

        instanceTypeName = instance.instanceType?.rawValue ??
        "<N/A>"
    }

    let instanceStateName: String
    if instanceState == nil {
        instanceStateName = "<N/A>"
    } else {
        instanceStateName = instanceState?.rawValue ?? "<N/
A>"
    }

    print("""
Instance: \(instance.instanceId ?? "<N/A>")
    • Image ID: \(instance.imageId ?? "<N/A>")
    • Instance type: \(instanceTypeName)
    • Key name: \(instance.keyName ?? "<N/A>")
    • VPC ID: \(instance.vpcId ?? "<N/A>")
    • Public IP: \(instance.publicIpAddress ?? "N/A")
    • State: \(instanceStateName)

""")

    return instance
    }
}
} catch {
    print("*** Error retrieving instance information to display:
\(error.localizedDescription)")
    return nil
}

return nil
}

/// Stop the specified instance.
///
/// - Parameters:
///   - instanceId: The ID of the instance to stop.
///   - waitUntilStopped: If `true`, execution waits until the instance
///     has stopped. Otherwise, execution continues and the instance stops
///     asynchronously.
///
/// - Returns: `true` if the image is successfully stopped (or is left to

```



```

    /// stop asynchronously). `false` if the instance doesn't stop.
    func stopInstance(instanceId: String, waitUntilStopped: Bool = false) async -
> Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.stopInstances(
                input: StopInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilStopped {
                print("Waiting for the instance to stop. Please be patient!")

                let waitOptions = WaiterOptions(maxWaitTime: 600)
                let output = try await ec2Client.waitUntilInstanceStopped(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )

                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to stop the instance:
\\(error.localizedDescription)")
            return false
        }
    }

    /// Start the specified instance.
    ///
    /// - Parameters:
    ///   - instanceId: The ID of the instance to start.
    ///   - waitUntilStarted: If `true`, execution waits until the instance

```

```
/// has started. Otherwise, execution continues and the instance starts
/// asynchronously.
///
/// - Returns: `true` if the image is successfully started (or is left to
/// start asynchronously). `false` if the instance doesn't start.
func startInstance(instanceId: String, waitUntilStarted: Bool = false) async
-> Bool {
    let instanceList = [instanceId]

    do {
        _ = try await ec2Client.startInstances(
            input: StartInstancesInput(
                instanceIds: instanceList
            )
        )

        if waitUntilStarted {
            print("Waiting for the instance to start...")

            let waitOptions = WaiterOptions(maxWaitTime: 60.0)
            let output = try await ec2Client.waitUntilInstanceRunning(
                options: waitOptions,
                input: DescribeInstancesInput(
                    instanceIds: instanceList
                )
            )
            switch output.result {
            case .success:
                return true
            case .failure:
                return false
            }
        } else {
            return true
        }
    } catch {
        print("*** Unable to start the instance:
\\(error.localizedDescription)")
        return false
    }
}

/// Terminate the specified instance.
///
```

```
/// - Parameters:
///   - instanceId: The instance to terminate.
///   - waitUntilTerminated: Whether or not to wait until the instance is
///     terminated before returning.
///
/// - Returns: `true` if terminated successfully. `false` if not or if an
///   error occurs.
func terminateInstance(instanceId: String, waitUntilTerminated: Bool = false)
async -> Bool {
    let instanceList = [instanceId]

    do {
        _ = try await ec2Client.terminateInstances(
            input: TerminateInstancesInput(
                instanceIds: instanceList
            )
        )

        if waitUntilTerminated {
            print("Waiting for the instance to terminate...")

            let waitOptions = WaiterOptions(maxWaitTime: 600.0)
            let output = try await ec2Client.waitUntilInstanceTerminated(
                options: waitOptions,
                input: DescribeInstancesInput(
                    instanceIds: instanceList
                )
            )

            switch output.result {
            case .success:
                return true
            case .failure:
                return false
            }
        } else {
            return true
        }
    } catch {
        print("*** Unable to terminate the instance:
\\(error.localizedDescription)")
        return false
    }
}
```

```

/// Return an array of `EC2ClientTypes.Image` objects describing all of
/// the images in the specified array.
///
/// - Parameter idList: A list of image ID strings indicating the images
///   to return details about.
///
/// - Returns: An array of the images.
func describeImages(_ idList: [String]) async -> [EC2ClientTypes.Image] {
    do {
        let output = try await ec2Client.describeImages(
            input: DescribeImagesInput(
                imageIds: idList
            )
        )

        guard let images = output.images else {
            print("*** No images found.")
            return []
        }

        for image in images {
            guard let id = image.imageId else {
                continue
            }
            print("    \(id): \(image.description ?? "<no description>")")
        }

        return images
    } catch {
        print("*** Error getting image descriptions:
\((error.localizedDescription)")
        return []
    }
}

/// Create and return a new EC2 instance.
///
/// - Parameters:
///   - imageId: The image ID of the AMI to use when creating the instance.
///   - instanceType: The type of instance to create.
///   - keyPairName: The RSA key pair's name to use to secure the instance.
///   - securityGroups: The security group or groups to add the instance
///     to.

```

```

///
/// - Returns: The EC2 instance as an `EC2ClientTypes.Instance` object.
func runInstance(imageId: String, instanceType: EC2ClientTypes.InstanceType,
                 keyPairName: String, securityGroups: [String]?) async ->
EC2ClientTypes.Instance? {
    do {
        let output = try await ec2Client.runInstances(
            input: RunInstancesInput(
                imageId: imageId,
                instanceType: instanceType,
                keyName: keyPairName,
                maxCount: 1,
                minCount: 1,
                securityGroupIds: securityGroups
            )
        )

        guard let instances = output.instances else {
            print("*** Unable to create the instance.")
            return nil
        }

        return instances[0]
    } catch {
        print("*** Error creating the instance:
\\(error.localizedDescription)")
        return nil
    }
}

/// Return the device's external IP address.
///
/// - Returns: A string containing the device's IP address.
func getMyIPAddress() -> String? {
    guard let url = URL(string: "http://checkip.amazonaws.com") else {
        print("Couldn't create the URL")
        return nil
    }

    do {
        print("Getting the IP address...")
        return try String(contentsOf: url, encoding:
String.Encoding.utf8).trim()
    } catch {

```

```

        print("*** Unable to get your public IP address.")
        return nil
    }
}

/// Create a new security group.
///
/// - Parameters:
///   - groupName: The name of the group to create.
///   - groupDescription: A description of the new security group.
///
/// - Returns: The ID string of the new security group.
func createSecurityGroup(name groupName: String, description
groupDescription: String) async -> String? {
    do {
        let output = try await ec2Client.createSecurityGroup(
            input: CreateSecurityGroupInput(
                description: groupDescription,
                groupName: groupName
            )
        )

        return output.groupId
    } catch {
        print("*** Error creating the security group:
\\(error.localizedDescription)")
        return nil
    }
}

/// Authorize ingress of connections for the security group.
///
/// - Parameters:
///   - groupId: The group ID of the security group to authorize access for.
///   - ipAddress: The IP address of the device to grant access to.
///
/// - Returns: `true` if access is successfully granted; otherwise `false`.
func authorizeSecurityGroupIngress(groupId: String, ipAddress: String) async
-> Bool {
    let ipRange = EC2ClientTypes.IpRange(cidrIp: "\\(ipAddress)/0")
    let httpPermission = EC2ClientTypes.IpPermission(
        fromPort: 80,
        ipProtocol: "tcp",
        ipRanges: [ipRange],

```

```

        toPort: 80
    )

    let sshPermission = EC2ClientTypes.IpPermission(
        fromPort: 22,
        ipProtocol: "tcp",
        ipRanges: [ipRange],
        toPort: 22
    )

    do {
        _ = try await ec2Client.authorizeSecurityGroupIngress(
            input: AuthorizeSecurityGroupIngressInput(
                groupId: groupId,
                ipPermissions: [httpPermission, sshPermission]
            )
        )

        return true
    } catch {
        print("*** Error authorizing ingress for the security group:
\\(error.localizedDescription)")
        return false
    }
}

func describeSecurityGroups(groupId: String) async -> Bool {
    do {
        let output = try await ec2Client.describeSecurityGroups(
            input: DescribeSecurityGroupsInput(
                groupIds: [groupId]
            )
        )

        guard let securityGroups = output.securityGroups else {
            print("No security groups found.")
            return true
        }

        for group in securityGroups {
            print("Group \\(group.groupId ?? "<unknown>") found with VPC
\\(group.vpcId ?? "<unknown>")")
        }
        return true
    }
}

```

```
    } catch {
        print("*** Error getting security group details:
\\(error.localizedDescription)")
        return false
    }
}

/// Delete a security group.
///
/// - Parameter groupId: The ID of the security group to delete.
///
/// - Returns: `true` on successful deletion; `false` on error.
func deleteSecurityGroup(groupId: String) async -> Bool {
    do {
        _ = try await ec2Client.deleteSecurityGroup(
            input: DeleteSecurityGroupInput(
                groupId: groupId
            )
        )

        return true
    } catch {
        print("*** Error deleting the security group:
\\(error.localizedDescription)")
        return false
    }
}

/// Allocate an Elastic IP address.
///
/// - Returns: A string containing the ID of the Elastic IP.
func allocateAddress() async -> String? {
    do {
        let output = try await ec2Client.allocateAddress(
            input: AllocateAddressInput(
                domain: EC2ClientTypes.DomainType.vpc
            )
        )

        guard let allocationId = output.allocationId else {
            return nil
        }

        return allocationId
    }
}
```



```

        } catch {
            print("*** Unable to allocate the IP address:
\(error.localizedDescription)")
            return nil
        }
    }

    /// Associate the specified allocated Elastic IP to a given instance.
    ///
    /// - Parameters:
    ///   - instanceId: The instance to associate the Elastic IP with.
    ///   - allocationId: The ID of the allocated Elastic IP to associate with
    ///     the instance.
    ///
    /// - Returns: The association ID of the association.
    func associateAddress(instanceId: String?, allocationId: String?) async ->
String? {
    do {
        let output = try await ec2Client.associateAddress(
            input: AssociateAddressInput(
                allocationId: allocationId,
                instanceId: instanceId
            )
        )

        return output.associationId
    } catch {
        print("*** Unable to associate the IP address:
\(error.localizedDescription)")
        return nil
    }
}

    /// Disassociate an Elastic IP.
    ///
    /// - Parameter associationId: The ID of the association to end.
    func disassociateAddress(associationId: String?) async {
        do {
            _ = try await ec2Client.disassociateAddress(
                input: DisassociateAddressInput(
                    associationId: associationId
                )
            )
        } catch {

```

```

        print("*** Unable to disassociate the IP address:
\(error.localizedDescription)")
    }
}

/// Release an allocated Elastic IP.
///
/// - Parameter allocationId: The allocation ID of the Elastic IP to
///   release.
func releaseAddress(allocationId: String?) async {
    do {
        _ = try await ec2Client.releaseAddress(
            input: ReleaseAddressInput(
                allocationId: allocationId
            )
        )
    } catch {
        print("*** Unable to release the IP address:
\(error.localizedDescription)")
    }
}

/// Generate and return a unique file name that begins with the specified
/// string.
///
/// - Parameters:
///   - prefix: Text to use at the beginning of the returned name.
///
/// - Returns: A string containing a unique filename that begins with the
///   specified `prefix`.
///
/// The returned name uses a random number between 1 million and 1 billion to
/// provide reasonable certainty of uniqueness for the purposes of this
/// example.
func tempName(prefix: String) -> String {
    return "\(prefix)-\(Int.random(in: 1000000..<1000000000))"
}

}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())
    }
}

```

```
        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- 有关 API 详细信息，请参阅《AWS SDK for Swift API Reference》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 SDK 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。

Amazon EC2 使用的操作 AWS SDKs

以下代码示例演示了如何使用执行各个 Amazon EC2 操作 AWS SDKs。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。

这些摘录调用 Amazon EC2 API，是大型程序的代码摘录，这些程序必须在上下文中运行。您可以在[Amazon EC2 使用的场景 AWS SDKs](#)中结合上下文查看操作。

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 API 参考](#)。

示例

- [将 AcceptVpcPeeringConnection 与 CLI 配合使用](#)
- [AllocateAddress 与 AWS SDK 或 CLI 配合使用](#)
- [将 AllocateHosts 与 CLI 配合使用](#)
- [将 AssignPrivateIpAddresses 与 CLI 配合使用](#)
- [AssociateAddress 与 AWS SDK 或 CLI 配合使用](#)
- [将 AssociateDhcpOptions 与 CLI 配合使用](#)
- [将 AssociateRouteTable 与 CLI 配合使用](#)
- [将 AttachInternetGateway 与 CLI 配合使用](#)
- [将 AttachNetworkInterface 与 CLI 配合使用](#)
- [将 AttachVolume 与 CLI 配合使用](#)
- [将 AttachVpnGateway 与 CLI 配合使用](#)
- [将 AuthorizeSecurityGroupEgress 与 CLI 配合使用](#)
- [AuthorizeSecurityGroupIngress 与 AWS SDK 或 CLI 配合使用](#)
- [将 CancelCapacityReservation 与 CLI 配合使用](#)
- [将 CancellImportTask 与 CLI 配合使用](#)
- [将 CancelSpotFleetRequests 与 CLI 配合使用](#)
- [将 CancelSpotInstanceRequests 与 CLI 配合使用](#)
- [将 ConfirmProductInstance 与 CLI 配合使用](#)
- [将 CopyImage 与 CLI 配合使用](#)
- [将 CopySnapshot 与 CLI 配合使用](#)

- [将 CreateCapacityReservation 与 CLI 配合使用](#)
- [将 CreateCustomerGateway 与 CLI 配合使用](#)
- [将 CreateDhcpOptions 与 CLI 配合使用](#)
- [将 CreateFlowLogs 与 CLI 配合使用](#)
- [将 CreateImage 与 CLI 配合使用](#)
- [将 CreateInstanceExportTask 与 CLI 配合使用](#)
- [将 CreateInternetGateway 与 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](#)
- [将 CreateNetworkAcl 与 CLI 配合使用](#)
- [将 CreateNetworkAclEntry 与 CLI 配合使用](#)
- [将 CreateNetworkInterface 与 CLI 配合使用](#)
- [将 CreatePlacementGroup 与 CLI 配合使用](#)
- [将 CreateRoute 与 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](#)
- [将 CreateSnapshot 与 CLI 配合使用](#)
- [将 CreateSpotDatafeedSubscription 与 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](#)
- [将 CreateVolume 与 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](#)
- [将 CreateVpnConnection 与 CLI 配合使用](#)
- [将 CreateVpnConnectionRoute 与 CLI 配合使用](#)
- [将 CreateVpnGateway 与 CLI 配合使用](#)
- [将 DeleteCustomerGateway 与 CLI 配合使用](#)
- [将 DeleteDhcpOptions 与 CLI 配合使用](#)

- [将 DeleteFlowLogs 与 CLI 配合使用](#)
- [将 DeleteInternetGateway 与 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](#)
- [将 DeleteNetworkAcl 与 CLI 配合使用](#)
- [将 DeleteNetworkAclEntry 与 CLI 配合使用](#)
- [将 DeleteNetworkInterface 与 CLI 配合使用](#)
- [将 DeletePlacementGroup 与 CLI 配合使用](#)
- [将 DeleteRoute 与 CLI 配合使用](#)
- [将 DeleteRouteTable 与 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](#)
- [将 DeleteSpotDatafeedSubscription 与 CLI 配合使用](#)
- [将 DeleteSubnet 与 CLI 配合使用](#)
- [将 DeleteTags 与 CLI 配合使用](#)
- [将 DeleteVolume 与 CLI 配合使用](#)
- [DeleteVpc与 AWS SDK 或 CLI 配合使用](#)
- [DeleteVpcEndpoints与 AWS SDK 或 CLI 配合使用](#)
- [将 DeleteVpnConnection 与 CLI 配合使用](#)
- [将 DeleteVpnConnectionRoute 与 CLI 配合使用](#)
- [将 DeleteVpnGateway 与 CLI 配合使用](#)
- [将 DeregisterImage 与 CLI 配合使用](#)
- [将 DescribeAccountAttributes 与 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](#)
- [将 DescribeBundleTasks 与 CLI 配合使用](#)
- [将 DescribeCapacityReservations 与 CLI 配合使用](#)
- [将 DescribeCustomerGateways 与 CLI 配合使用](#)

- [将 DescribeDhcpOptions 与 CLI 配合使用](#)
- [将 DescribeFlowLogs 与 CLI 配合使用](#)
- [将 DescribeHostReservationOfferings 与 CLI 配合使用](#)
- [将 DescribeHosts 与 CLI 配合使用](#)
- [DescribeIamInstanceProfileAssociations与 AWS SDK 或 CLI 配合使用](#)
- [将 DescribeIdFormat 与 CLI 配合使用](#)
- [将 DescribeIdentityIdFormat 与 CLI 配合使用](#)
- [将 DescribeImageAttribute 与 CLI 配合使用](#)
- [DescribeImages与 AWS SDK 或 CLI 配合使用](#)
- [将 DescribeImportImageTasks 与 CLI 配合使用](#)
- [将 DescribeImportSnapshotTasks 与 CLI 配合使用](#)
- [将 DescribeInstanceAttribute 与 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](#)
- [将 DescribeInternetGateways 与 CLI 配合使用](#)
- [DescribeKeyPairs与 AWS SDK 或 CLI 配合使用](#)
- [将 DescribeNetworkAcls 与 CLI 配合使用](#)
- [将 DescribeNetworkInterfaceAttribute 与 CLI 配合使用](#)
- [将 DescribeNetworkInterfaces 与 CLI 配合使用](#)
- [将 DescribePlacementGroups 与 CLI 配合使用](#)
- [将 DescribePrefixLists 与 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](#)
- [将 DescribeScheduledInstanceAvailability 与 CLI 配合使用](#)
- [将 DescribeScheduledInstances 与 CLI 配合使用](#)
- [DescribeSecurityGroups与 AWS SDK 或 CLI 配合使用](#)
- [将 DescribeSnapshotAttribute 与 CLI 配合使用](#)
- [DescribeSnapshots与 AWS SDK 或 CLI 配合使用](#)

- [将 DescribeSpotDatafeedSubscription 与 CLI 配合使用](#)
- [将 DescribeSpotFleetInstances 与 CLI 配合使用](#)
- [将 DescribeSpotFleetRequestHistory 与 CLI 配合使用](#)
- [将 DescribeSpotFleetRequests 与 CLI 配合使用](#)
- [将 DescribeSpotInstanceRequests 与 CLI 配合使用](#)
- [将 DescribeSpotPriceHistory 与 CLI 配合使用](#)
- [DescribeSubnets 与 AWS SDK 或 CLI 配合使用](#)
- [将 DescribeTags 与 CLI 配合使用](#)
- [将 DescribeVolumeAttribute 与 CLI 配合使用](#)
- [将 DescribeVolumeStatus 与 CLI 配合使用](#)
- [将 DescribeVolumes 与 CLI 配合使用](#)
- [将 DescribeVpcAttribute 与 CLI 配合使用](#)
- [将 DescribeVpcClassicLink 与 CLI 配合使用](#)
- [将 DescribeVpcClassicLinkDnsSupport 与 CLI 配合使用](#)
- [将 DescribeVpcEndpointServices 与 CLI 配合使用](#)
- [将 DescribeVpcEndpoints 与 CLI 配合使用](#)
- [DescribeVpcs 与 AWS SDK 或 CLI 配合使用](#)
- [将 DescribeVpnConnections 与 CLI 配合使用](#)
- [将 DescribeVpnGateways 与 CLI 配合使用](#)
- [将 DetachInternetGateway 与 CLI 配合使用](#)
- [将 DetachNetworkInterface 与 CLI 配合使用](#)
- [将 DetachVolume 与 CLI 配合使用](#)
- [将 DetachVpnGateway 与 CLI 配合使用](#)
- [将 DisableVgwRoutePropagation 与 CLI 配合使用](#)
- [将 DisableVpcClassicLink 与 CLI 配合使用](#)
- [将 DisableVpcClassicLinkDnsSupport 与 CLI 配合使用](#)
- [DisassociateAddress 与 AWS SDK 或 CLI 配合使用](#)
- [将 DisassociateRouteTable 与 CLI 配合使用](#)

- [将 EnableVgwRoutePropagation 与 CLI 配合使用](#)
- [将 EnableVolumelo 与 CLI 配合使用](#)
- [将 EnableVpcClassicLink 与 CLI 配合使用](#)
- [将 EnableVpcClassicLinkDnsSupport 与 CLI 配合使用](#)
- [将 GetConsoleOutput 与 CLI 配合使用](#)
- [将 GetHostReservationPurchasePreview 与 CLI 配合使用](#)
- [GetPasswordData与 AWS SDK 或 CLI 配合使用](#)
- [将 ImportImage 与 CLI 配合使用](#)
- [将 ImportKeyPair 与 CLI 配合使用](#)
- [将 ImportSnapshot 与 CLI 配合使用](#)
- [将 ModifyCapacityReservation 与 CLI 配合使用](#)
- [将 ModifyHosts 与 CLI 配合使用](#)
- [将 ModifyIdFormat 与 CLI 配合使用](#)
- [将 ModifyImageAttribute 与 CLI 配合使用](#)
- [将 ModifyInstanceAttribute 与 CLI 配合使用](#)
- [将 ModifyInstanceCreditSpecification 与 CLI 配合使用](#)
- [将 ModifyNetworkInterfaceAttribute 与 CLI 配合使用](#)
- [将 ModifyReservedInstances 与 CLI 配合使用](#)
- [将 ModifySnapshotAttribute 与 CLI 配合使用](#)
- [将 ModifySpotFleetRequest 与 CLI 配合使用](#)
- [将 ModifySubnetAttribute 与 CLI 配合使用](#)
- [将 ModifyVolumeAttribute 与 CLI 配合使用](#)
- [将 ModifyVpcAttribute 与 CLI 配合使用](#)
- [MonitorInstances与 AWS SDK 或 CLI 配合使用](#)
- [将 MoveAddressToVpc 与 CLI 配合使用](#)
- [将 PurchaseHostReservation 与 CLI 配合使用](#)
- [将 PurchaseScheduledInstances 与 CLI 配合使用](#)
- [RebootInstances与 AWS SDK 或 CLI 配合使用](#)
- [将 RegisterImage 与 CLI 配合使用](#)

- [将 RejectVpcPeeringConnection 与 CLI 配合使用](#)
- [ReleaseAddress 与 AWS SDK 或 CLI 配合使用](#)
- [将 ReleaseHosts 与 CLI 配合使用](#)
- [ReplacelamInstanceProfileAssociation 与 AWS SDK 或 CLI 配合使用](#)
- [将 ReplaceNetworkAclAssociation 与 CLI 配合使用](#)
- [将 ReplaceNetworkAclEntry 与 CLI 配合使用](#)
- [将 ReplaceRoute 与 CLI 配合使用](#)
- [将 ReplaceRouteTableAssociation 与 CLI 配合使用](#)
- [将 ReportInstanceState 与 CLI 配合使用](#)
- [将 RequestSpotFleet 与 CLI 配合使用](#)
- [将 RequestSpotInstances 与 CLI 配合使用](#)
- [将 ResetImageAttribute 与 CLI 配合使用](#)
- [将 ResetInstanceAttribute 与 CLI 配合使用](#)
- [将 ResetNetworkInterfaceAttribute 与 CLI 配合使用](#)
- [将 ResetSnapshotAttribute 与 CLI 配合使用](#)
- [将 RevokeSecurityGroupEgress 与 CLI 配合使用](#)
- [将 RevokeSecurityGroupIngress 与 CLI 配合使用](#)
- [RunInstances 与 AWS SDK 或 CLI 配合使用](#)
- [将 RunScheduledInstances 与 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](#)
- [将 UnassignPrivateIpAddresses 与 CLI 配合使用](#)
- [UnmonitorInstances 与 AWS SDK 或 CLI 配合使用](#)
- [将 UpdateSecurityGroupRuleDescriptionsIngress 与 CLI 配合使用](#)

将 **AcceptVpcPeeringConnection** 与 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": [],
    "AcceptorVpcInfo": {
      "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

适用于 PowerShell V4 的工具

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

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

输出：

```

AcceptorVpcInfo      : 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 \(V 4\)](#) [AcceptVpcPeeringConnection](#) 中的。

适用于 PowerShell V5 的工具

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

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

输出：

```

AcceptorVpcInfo      : 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 \(V 5\)](#) [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 SDK 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。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)
- [创建具有私有子网和 NAT 网关的 VPC](#)
- [开始使用 Amazon VPC](#)

.NET

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。

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

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

```

        }

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

```

- 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](#) 中查找完整示例，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."
  fi
}

```



```

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](#) 中查找完整示例，了解如何进行设置和运行。

```

//! Allocate an Elastic IP address and associate it with an Amazon Elastic
    Compute Cloud
    //! (Amazon EC2) instance.
    /*!
        \param instanceID: An EC2 instance ID.
        \param[out] publicIPAddress: String to return the public IP address.
        \param[out] allocationID: String to return the allocation ID.
        \param clientConfiguration: AWS client configuration.
        \return bool: Function succeeded.
    */
    bool AwsDoc::EC2::allocateAndAssociateAddress(const Aws::String &instanceId,
        Aws::String &publicIPAddress,

```

```
        Aws::String &allocationID,
        const
    Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::AllocateAddressRequest request;
    request.SetDomain(Aws::EC2::Model::DomainType::vpc);

    const Aws::EC2::Model::AllocateAddressOutcome outcome =
        ec2Client.AllocateAddress(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to allocate Elastic IP address:" <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }
    const Aws::EC2::Model::AllocateAddressResponse &response =
    outcome.GetResult();
    allocationID = response.GetAllocationId();
    publicIPAddress = response.GetPublicIp();

    return true;
}
```

- 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。亚马逊从亚马逊的地址池 EC2 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 从地址池 EC2 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](#)。

示例 4：从 IPAM 池中分配弹性 IP 地址

以下 `allocate-address` 示例从 Amazon VPC IP 地址管理器 (IPAM) 池中分配特定的 /32 弹性 IP 地址。

```
aws ec2 allocate-address \
  --region us-east-1 \
  --ipam-pool-id ipam-pool-1234567890abcdef0 \
  --address 192.0.2.0
```

输出：

```
{
  "PublicIp": "192.0.2.0",
  "AllocationId": "eipalloc-abcdef01234567890",
  "PublicIpv4Pool": "ipam-pool-1234567890abcdef0",
  "NetworkBorderGroup": "us-east-1",
  "Domain": "vpc"
}
```

有关更多信息，请参阅《Amazon VPC IPAM User Guide》中的 [Allocate sequential Elastic IP addresses from an IPAM pool](#)。

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```

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

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

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```

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

/**
 * Allocates an Elastic IP address to your AWS account.
 */

```

```

export const main = async () => {
  const client = new EC2Client({});
  const command = new AllocateAddressCommand({});

  try {
    const { AllocationId, PublicIp } = await client.send(command);
    console.log("A new IP address has been allocated to your account:");
    console.log(`ID: ${AllocationId} Public IP: ${PublicIp}`);
    console.log(
      "You can view your IP addresses in the AWS Management Console for Amazon
      EC2. Look under Network & Security > Elastic IPs",
    );
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide these values?`);
    } else {
      throw caught;
    }
  }
};
import { fileURLToPath } from "node:url";
// Call function if run directly.
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  main();
}

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```

suspend fun getAllocateAddress(instanceIdVal: String?): String? {
    val allocateRequest =
        AllocateAddressRequest {

```

```

        domain = DomainType.Vpc
    }

    Ec2Client.fromEnvironment { 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 Kotlin 的 AWS SDK API 参考。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例分配一个弹性 IP 地址，供 VPC 中的实例使用。

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

输出：

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

示例 2：此示例分配弹性 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 \(V 4\)](#) [AllocateAddress](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例分配一个弹性 IP 地址，供 VPC 中的实例使用。

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

输出：

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

示例 2：此示例分配弹性 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 \(V 5\)](#) [AllocateAddress](#) 中的。

Python

适用于 Python 的 SDK (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。


```

class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
    actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
            = None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

        def __init__(self, ec2_client: Any) -> None:
            """
            Initializes the ElasticIpWrapper with an EC2 client.

            :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
            level
                               access to AWS EC2 services.
            """
            self.ec2_client = ec2_client
            self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
        client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

```

```
def allocate(self) -> "ElasticIpWrapper.ElasticIp":
    """
    Allocates an Elastic IP address that can be associated with an Amazon EC2
    instance. By using an Elastic IP address, you can keep the public IP
    address
    constant even when you restart the associated instance.

    :return: The ElasticIp object for the newly created Elastic IP address.
    :raises ClientError: If the allocation fails, such as reaching the
    maximum limit of Elastic IPs.
    """
    try:
        response = self.ec2_client.allocate_address(Domain="vpc")
        elastic_ip = self.ElasticIp(
            allocation_id=response["AllocationId"],
            public_ip=response["PublicIp"]
        )
        self.elastic_ips.append(elastic_ip)
    except ClientError as err:
        if err.response["Error"]["Code"] == "AddressLimitExceeded":
            logger.error(
                "Max IP's reached. Release unused addresses or contact AWS
                Support for an increase."
            )
        raise err
    return elastic_ip
```

- 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](#) 中查找完整示例，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')
  response.allocation_id
rescue StandardError => e
  puts "Error allocating Elastic IP address: #{e.message}"
  'Error'
end
```

- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考 [AllocateAddress](#) 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](#) 中查找完整示例，了解如何进行设置和运行。

```
pub async fn allocate_ip_address(&self) -> Result<AllocateAddressOutput,
EC2Error> {
    self.client
```

```

        .allocate_address()
        .domain(DomainType::Vpc)
        .send()
        .await
        .map_err(EC2Error::from)
    }

```

- 有关 API 的详细信息，请参阅适用[AllocateAddress](#)于 Rust 的AWS SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，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。

Swift

适用于 Swift 的 SDK

Note

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

```
import AWSEC2

/// Allocate an Elastic IP address.
///
/// - Returns: A string containing the ID of the Elastic IP.
func allocateAddress() async -> String? {
    do {
        let output = try await ec2Client.allocateAddress(
            input: AllocateAddressInput(
                domain: EC2ClientTypes.DomainType.vpc
            )
        )

        guard let allocationId = output.allocationId else {
            return nil
        }

        return allocationId
    } catch {
        print("*** Unable to allocate the IP address:
\\(error.localizedDescription)")
        return nil
    }
}
```

- 有关 API 的详细信息，请参阅适用于 Swift 的 AWS SDK API 参考 [AllocateAddress](#) 中。

有关 AWS SDK 开发者指南和代码示例的完整列表，请参阅 [使用 AWS SDK 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。

将 `AllocateHosts` 与 CLI 配合使用

以下代码示例演示如何使用 `AllocateHosts`。

CLI

AWS CLI

示例 1：分配专属主机

以下 `allocate-hosts` 示例在 `eu-west-1a` 可用区中分配一个专属主机，您可以在其上启动 `m5.large` 实例。默认情况下，专属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：分配启用了自动置放和主机恢复的专属主机

以下 `allocate-hosts` 示例在启用了自动置放和主机恢复的 `eu-west-1a` 可用区中分配单个专属主机。

```
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：分配带有标签的专属主机

以下 `allocate-hosts` 示例分配单个专属主机，并应用具有名为 `purpose` 的键和值为 `production` 的标签。

```
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 Amazon EC2 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

适用于 PowerShell V4 的工具

示例 1：此示例向您的账户分配给定实例类型和可用区的专属主机

```
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 (V 4) [AllocateHosts](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例向您的账户分配给定实例类型和可用区的专属主机

```
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 (V 5) [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 SDK 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。

将 **AssignPrivateIpAddresses** 与 CLI 配合使用

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

CLI

AWS CLI

为网络接口分配特定的辅助私有 IP 地址

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

此示例将两个辅助私有 IP 地址分配给指定的网络接口。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

适用于 PowerShell V4 的工具

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 (V 4) [AssignPrivateIpAddresses](#)中的。

适用于 PowerShell V5 的工具

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 (V 5) [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 SDK 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：

- [了解基本功能](#)

.NET

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。

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

```

        {
            AllocationId = allocationId,
            InstanceId = instanceId
        };

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

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

```

- 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](#) 中查找完整示例，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
}

```

```

        ;;
    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](#) 中查找完整示例，了解如何进行设置和运行。

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

//! Associate an Elastic IP address with an EC2 instance.
/*!
    \param instanceId: An EC2 instance ID.
    \param allocationId: An Elastic IP allocation ID.
    \param[out] associationId: String to receive the association ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: True if the address was associated with the instance; otherwise,
    false.
*/
bool AwsDoc::EC2::associateAddress(const Aws::String &instanceId, const
    Aws::String &allocationId,
                                   Aws::String &associationId,
                                   const Aws::Client::ClientConfiguration
    &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

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

    Aws::EC2::Model::AssociateAddressOutcome outcome =
    ec2Client.AssociateAddress(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to associate address " << allocationId <<
            " with instance " << instanceId << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully associated address " << allocationId <<
            " with instance " << instanceId << std::endl;
    }
}
```

```
        associationID = outcome.GetResult().GetAssociationId();
    }

    return outcome.IsSuccess();
}
```

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

CLI

AWS CLI

示例 1：关联弹性 IP 地址和实例

以下 `associate-address` 示例将弹性 IP 地址与指定 EC2 实例相关联。

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

输出：

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

示例 2：将弹性 IP 地址与网络接口相关联

以下 `associate-address` 示例将指定的弹性 IP 地址与指定的网络接口相关联。

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

输出：

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



```
}
```

示例 3：将弹性 IP 地址与私有 IP 地址关联

以下 `associate-address` 示例将指定的弹性 IP 地址与指定网络接口中的指定私有 IP 地址相关联。

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

输出：

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

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[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](#) 中查找完整示例，了解如何进行设置和运行。

```
/**  
 * Associates an Elastic IP address with an EC2 instance asynchronously.  
 *  
 * @param instanceId    the ID of the EC2 instance to associate the Elastic  
 IP address with  
 * @param allocationId  the allocation ID of the Elastic IP address to  
 associate
```

```

    * @return a {@link CompletableFuture} that completes with the association ID
    when the operation is successful,
    *         or throws a {@link RuntimeException} if the operation fails
    */
    public CompletableFuture<String> associateAddressAsync(String instanceId,
String allocationId) {
        AssociateAddressRequest associateRequest =
AssociateAddressRequest.builder()
            .instanceId(instanceId)
            .allocationId(allocationId)
            .build();

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

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

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

```

/**
 * Associates an Elastic IP address, or carrier IP address (for instances that
 * are in subnets in Wavelength Zones)
 * with an instance or a network interface.
 * @param {{ instanceId: string, allocationId: string }} options
 */
export const main = async ({ instanceId, allocationId }) => {
  const client = new EC2Client({});
  const command = new AssociateAddressCommand({
    // You need to allocate an Elastic IP address before associating it with an
    instance.
    // You can do that with the AllocateAddressCommand.
    AllocationId: allocationId,
    // You need to create an EC2 instance before an IP address can be associated
    with it.
    // You can do that with the RunInstancesCommand.
    InstanceId: instanceId,
  });

  try {
    const { AssociationId } = await client.send(command);
    console.log(
      `Address with allocation ID ${allocationId} is now associated with instance
${instanceId}.`,
      `The association ID is ${AssociationId}.`,
    );
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAllocationID.NotFound"
    ) {
      console.warn(
        `${caught.message}. Did you provide the ID of a valid Elastic IP address
AllocationId?`,
      );
    } else {
      throw caught;
    }
  }
};

```

- 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](#) 中查找完整示例，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.fromEnvironment { 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 Kotlin 的 AWS SDK API 参考。

PowerShell

适用于 PowerShell V4 的工具

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 \(V 4\) AssociateAddress](#) 中的。

适用于 PowerShell V5 的工具

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 \(V 5\) AssociateAddress](#) 中的。

Python

适用于 Python 的 SDK (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。

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

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""
```

```

    def __init__(
        self, allocation_id: str, public_ip: str, instance_id: Optional[str]
= None
    ) -> None:
        """
        Initializes the ElasticIp object.

        :param allocation_id: The allocation ID of the Elastic IP.
        :param public_ip: The public IP address of the Elastic IP.
        :param instance_id: The ID of the associated EC2 instance, if any.
        """
        self.allocation_id = allocation_id
        self.public_ip = public_ip
        self.instance_id = instance_id

    def __init__(self, ec2_client: Any) -> None:
        """
        Initializes the ElasticIpWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
        self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def associate(
        self, allocation_id: str, instance_id: str
    ) -> Union[Dict[str, Any], None]:
        """

```

```

Associates an Elastic IP address with an instance. When this association
is
created, the Elastic IP's public IP address is immediately used as the
public
IP address of the associated instance.

:param allocation_id: The allocation ID of the Elastic IP.
:param instance_id: The ID of the Amazon EC2 instance.
:return: A response that contains the ID of the association, or None if
no Elastic IP is found.
:raises ClientError: If the association fails, such as when the instance
ID is not found.
"""
    elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
    if elastic_ip is None:
        logger.info(f"No Elastic IP found with allocation ID
{allocation_id}.")
        return None

    try:
        response = self.ec2_client.associate_address(
            AllocationId=allocation_id, InstanceId=instance_id
        )
        elastic_ip.instance_id = (
            instance_id # Track the instance associated with this Elastic
IP.
        )
    except ClientError as err:
        if err.response["Error"]["Code"] == "InvalidInstanceID.NotFound":
            logger.error(
                f"Failed to associate Elastic IP {allocation_id} with
{instance_id} "
                "because the specified instance ID does not exist or has not
propagated fully. "
                "Verify the instance ID and try again, or wait a few moments
before attempting to "
                "associate the Elastic IP address."
            )
            raise
    return response

```

- 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](#) 中查找完整示例，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 association ID corresponding to the association of the
#   Elastic IP address to the instance.
# @example
#   puts allocate_elastic_ip_address(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX',
#     'i-033c48ef067af3dEX')
def associate_elastic_ip_address_with_instance(
  ec2_client,
  allocation_id,
  instance_id
)
  response = ec2_client.associate_address(
    allocation_id: allocation_id,
    instance_id: instance_id
  )
end
```



```
response.association_id
rescue StandardError => e
  puts "Error associating Elastic IP address with instance: #{e.message}"
  'Error'
end
```

- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考[AssociateAddress](#)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](#) 中查找完整示例，了解如何进行设置和运行。

```
pub async fn associate_ip_address(
    &self,
    allocation_id: &str,
    instance_id: &str,
) -> Result<AssociateAddressOutput, EC2Error> {
    let response = self
        .client
        .associate_address()
        .allocation_id(allocation_id)
        .instance_id(instance_id)
        .send()
        .await?;
    Ok(response)
}
```

- 有关 API 的详细信息，请参阅适用[AssociateAddress](#)于 Rust 的AWS SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，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 SAP 的 AWS SDK ABAP API 参考。

Swift

适用于 Swift 的 SDK

 Note

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

```
import AWSEC2

/// Associate the specified allocated Elastic IP to a given instance.
///
```

```

    /// - Parameters:
    ///   - instanceId: The instance to associate the Elastic IP with.
    ///   - allocationId: The ID of the allocated Elastic IP to associate with
    ///     the instance.
    ///
    /// - Returns: The association ID of the association.
    func associateAddress(instanceId: String?, allocationId: String?) async ->
    String? {
        do {
            let output = try await ec2Client.associateAddress(
                input: AssociateAddressInput(
                    allocationId: allocationId,
                    instanceId: instanceId
                )
            )

            return output.associationId
        } catch {
            print("*** Unable to associate the IP address:
\\(error.localizedDescription)")
            return nil
        }
    }
}

```

- 有关 API 的详细信息，请参阅适用于 Swift 的 AWS SDK API 参考 [AssociateAddress](#) 中。

有关 AWS SDK 开发者指南和代码示例的完整列表，请参阅 [使用 AWS SDK 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。

将 **AssociateDhcpOptions** 与 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 VPC 相关联

此示例将指定的 DHCP 选项集与指定的 VPC 相关联。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 VPC 相关联

此示例将默认 DHCP 选项集与指定的 VPC 相关联。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

适用于 PowerShell V4 的工具

示例 1：此示例将指定的 DHCP 选项集与指定的 VPC 相关联。

```
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 相关联。

```
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 (V 4) [AssociateDhcpOptions](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例将指定的 DHCP 选项集与指定的 VPC 相关联。

```
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 相关联。

```
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 (V 5) [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 SDK 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。

将 **AssociateRouteTable** 与 CLI 配合使用

以下代码示例演示如何使用 AssociateRouteTable。

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：

- [创建具有私有子网和 NAT 网关的 VPC](#)
- [开始使用 Amazon VPC](#)

CLI

AWS CLI

将路由表与子网相关联

此示例将指定的路由表与指定的子网相关联。

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

适用于 PowerShell V4 的工具

示例 1：此示例将指定的路由表与指定的子网相关联。

```
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 \(V 4\)](#) [AssociateRouteTable](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例将指定的路由表与指定的子网相关联。

```
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 \(V 5\)](#) [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 SDK 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。

将 **AttachInternetGateway** 与 CLI 配合使用

以下代码示例演示如何使用 `AttachInternetGateway`。

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：

- [创建具有私有子网和 NAT 网关的 VPC](#)
- [开始使用 Amazon VPC](#)

CLI

AWS CLI

将互联网网关连接到 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

适用于 PowerShell V4 的工具

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 (V 4) [AttachInternetGateway](#) 中的。

适用于 PowerShell V5 的工具

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 (V 5) [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 SDK 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。

将 `AttachNetworkInterface` 与 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：将网络接口连接到具有多个网卡的实例

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

适用于 PowerShell V4 的工具

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 (V 4) [AttachNetworkInterface](#)中的。

适用于 PowerShell V5 的工具

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 (V 5) [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 SDK 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。

将 **AttachVolume** 与 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) 作为 /dev/sdf 连接到实例 (i-01474ef662b89480)。

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

适用于 PowerShell V4 的工具

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公开它。

```
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 (V 4) [AttachVolume](#) 中的。

适用于 PowerShell V5 的工具

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公开它。

```
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 (V 5) [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 SDK 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。

将 **AttachVpnGateway** 与 CLI 配合使用

以下代码示例演示如何使用 **AttachVpnGateway**。

CLI

AWS CLI

将虚拟专用网关连接到 VPC

以下 `attach-vpn-gateway` 示例将指定的虚拟专用网关连接到指定的 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

适用于 PowerShell V4 的工具

示例 1：此示例将指定的虚拟专用网关连接到指定的 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 (V 4) [AttachVpnGateway](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例将指定的虚拟专用网关连接到指定的 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 \(V 5\) 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 SDK 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。

将 `AuthorizeSecurityGroupEgress` 与 CLI 配合使用

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

CLI

AWS CLI

示例 1：添加允许指向特定地址范围的出站流量的规则

以下 `authorize-security-group-egress` 示例添加授予对 TCP 端口 80 上指定地址范围的访问权限的规则。

```
aws ec2 authorize-security-group-egress \
  --group-id sg-1234567890abcdef0 \
  --ip-permissions
  'IpProtocol=tcp,FromPort=80,ToPort=80,IpRanges=[{CidrIp=10.0.0.0/16}]'
```

输出：

```
{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupRuleId": "sgr-0b15794cdb17bf29c",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "IsEgress": true,
      "IpProtocol": "tcp",
      "FromPort": 80,
```

```

        "ToPort": 80,
        "CidrIpv4": "10.0.0.0/16"
    }
]
}

```

示例 2：添加允许指向特定安全组的出站流量的规则

以下 `authorize-security-group-egress` 示例添加授予对 TCP 端口 80 上指定安全组的访问权限的规则。

```

aws ec2 authorize-security-group-egress \
  --group-id sg-1234567890abcdef0 \
  --ip-permissions
'IpProtocol=tcp,FromPort=80,ToPort=80,UserIdGroupPairs=[{GroupId=sg-0aad1c26bbeec5c22}]'

```

输出：

```

{
  "Return": true,
  "SecurityGroupRules": [
    {
      "SecurityGroupRuleId": "sgr-0b5dd815afcea9cc3",
      "GroupId": "sg-1234567890abcdef0",
      "GroupOwnerId": "123456789012",
      "IsEgress": true,
      "IpProtocol": "tcp",
      "FromPort": 80,
      "ToPort": 80,
      "ReferencedGroupInfo": {
        "GroupId": "sg-0aad1c26bbeec5c22",
        "UserId": "123456789012"
      }
    }
  ]
}

```

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[AuthorizeSecurityGroupEgress](#)中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例为 EC2-VPC 的指定安全组定义了出口规则。此规则授予对 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：此示例授予对 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"  
  
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 \(V 4\) AuthorizeSecurityGroupEgress](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例为 EC2-VPC 的指定安全组定义了出口规则。此规则授予对 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：此示例授予对 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"

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 (V 5) [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 SDK 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。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)
- [创建具有私有子网和 NAT 网关的 VPC](#)
- [开始使用 Amazon VPC](#)

.NET

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。

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

```

```

        _logger.LogError(
            $"The ingress rule already exists. {ec2Exception.Message}");
    }

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

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

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

```

- 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](#) 中查找完整示例，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 getopt command in a function.

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

```

}

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

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

if [[ -z "$ip_address" ]]; then
    errecho "ERROR: You must provide an IP address or CIDR block with the -i
parameter."
    usage
    return 1
fi

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

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

```

```

        usage
        return 1
    fi

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

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

    return 0
}

```

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](#) 中查找完整示例，了解如何进行设置和运行。

```
#!/ Authorize ingress to an Amazon Elastic Compute Cloud (Amazon EC2) group.
```

```

/!*
 \param groupID: The EC2 group ID.
 \param clientConfiguration: The ClientConfiguration object.
 \return bool: True if the operation was successful, false otherwise.
 */
bool
AwsDoc::EC2::authorizeSecurityGroupIngress(const Aws::String &groupID,
                                           const Aws::Client::ClientConfiguration
                                           &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::AuthorizeSecurityGroupIngressRequest
authorizeSecurityGroupIngressRequest;
    authorizeSecurityGroupIngressRequest.SetGroupId(groupID);
    buildSampleIngressRule(authorizeSecurityGroupIngressRequest);

    Aws::EC2::Model::AuthorizeSecurityGroupIngressOutcome
authorizeSecurityGroupIngressOutcome =

ec2Client.AuthorizeSecurityGroupIngress(authorizeSecurityGroupIngressRequest);

    if (authorizeSecurityGroupIngressOutcome.IsSuccess()) {
        std::cout << "Successfully authorized security group ingress." <<
std::endl;
    } else {
        std::cerr << "Error authorizing security group ingress: "
        << authorizeSecurityGroupIngressOutcome.GetError().GetMessage()
<< std::endl;
    }

    return authorizeSecurityGroupIngressOutcome.IsSuccess();
}

```

用于构建入口规则的实用程序函数。

```

//! Build a sample ingress rule.
/!*
 \param authorize_request: An 'AuthorizeSecurityGroupIngressRequest' instance.
 \return void:
 */
void buildSampleIngressRule(
    Aws::EC2::Model::AuthorizeSecurityGroupIngressRequest &authorize_request)
{

```

```

    Aws::String ingressIPRange = "203.0.113.0/24"; // Configure this for your
    allowed IP range.
    Aws::EC2::Model::IpRange ip_range;
    ip_range.SetCidrIp(ingressIPRange);

    Aws::EC2::Model::IpPermission permission1;
    permission1.SetIpProtocol("tcp");
    permission1.SetToPort(80);
    permission1.SetFromPort(80);
    permission1.AddIpRanges(ip_range);

    authorize_request.AddIpPermissions(permission1);

    Aws::EC2::Model::IpPermission permission2;
    permission2.SetIpProtocol("tcp");
    permission2.SetToPort(22);
    permission2.SetFromPort(22);
    permission2.AddIpRanges(ip_range);

    authorize_request.AddIpPermissions(permission2);
}

```

- 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
  'IpProtocol=tcp,FromPort=3389,ToPort=3389,IpRanges=[{CidrIp=172.31.0.0/16}]'
  'IpProtocol=icmp,FromPort=-1,ToPort=-1,IpRanges=[{CidrIp=172.31.0.0/16}]'

```

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
  'IpProtocol=icmp,FromPort=3,ToPort=4,IpRanges=[{CidrIp=0.0.0.0/0}]'
```

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：添加 IPv6 流量规则

以下 `authorize-security-group-ingress` 示例使用 `ip-permissions` 参数添加一条入站规则，该规则允许从该 IPv6 范围进行 SSH 访问（端口 22）`2001:db8:1234:1a00::/64`。

```
aws ec2 authorize-security-group-ingress \
  --group-id sg-1234567890abcdef0 \
  --ip-permissions
  'IpProtocol=tcp,FromPort=22,ToPort=22,Ipv6Ranges=[{CidrIpv6=2001:db8:1234:1a00::/64}]'
```

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：添加 ICMPv6 流量规则

以下 `authorize-security-group-ingress` 示例使用 `ip-permissions` 参数添加允许来自任何地方的 ICMPv6 流量的入站规则。

```
aws ec2 authorize-security-group-ingress \
  --group-id sg-1234567890abcdef0 \
  --ip-permissions 'IpProtocol=icmpv6,Ipv6Ranges=[{CidrIpv6=::/0}]'
```

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
  'IpProtocol=tcp,FromPort=3389,ToPort=3389,IpRanges=[{CidrIp=203.0.113.0/24,Description='
  office'}]'
```

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
  'IpProtocol=all,PrefixListIds=[{PrefixListId=pl-002dc3ec097de1514}]'
```

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](#) 中查找完整示例，了解如何进行设置和运行。

```
/**
 * Creates a new security group asynchronously with the specified group name,
 * description, and VPC ID. It also
 * authorizes inbound traffic on ports 80 and 22 from the specified IP
 * address.
 *
 * @param groupName the name of the security group to create
 * @param groupDesc the description of the security group
 * @param vpcId the ID of the VPC in which to create the security
 * group
```

```

    * @param myIpAddress the IP address from which to allow inbound traffic
    (e.g., "192.168.1.1/0" to allow traffic from
    *
    * any IP address in the 192.168.1.0/24 subnet)
    * @return a CompletableFuture that, when completed, returns the ID of the
    created security group
    * @throws RuntimeException if there was a failure creating the security
    group or authorizing the inbound traffic
    */
    public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

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

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

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

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

```

```

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

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```

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

/**
 * Adds the specified inbound (ingress) rules to a security group.
 * @param {{ groupId: string, ipAddress: string }} options
 */
export const main = async ({ groupId, ipAddress }) => {

```



```
const client = new EC2Client({});
const command = new AuthorizeSecurityGroupIngressCommand({
  // Use a group ID from the AWS console or
  // the DescribeSecurityGroupsCommand.
  GroupId: groupId,
  IpPermissions: [
    {
      IpProtocol: "tcp",
      FromPort: 22,
      ToPort: 22,
      // The IP address to authorize.
      // For more information on this notation, see
      // https://en.wikipedia.org/wiki/Classless_Inter-
Domain_Routing#CIDR_notation
      IpRanges: [{ CidrIp: `${ipAddress}/32` }],
    },
  ],
});

try {
  const { SecurityGroupRules } = await client.send(command);
  console.log(JSON.stringify(SecurityGroupRules, null, 2));
} catch (caught) {
  if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
    console.warn(`${caught.message}. Please provide a valid GroupId.`);
  } else {
    throw caught;
  }
}
};
```

- 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](#) 中查找完整示例，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.fromEnvironment { 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 Kotlin 的 AWS SDK API 参考。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例定义了 EC2-VPC 安全组的入口规则。这些规则为 SSH (端口 22) 和 RDC (端口 3389) 授予对特定 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：此示例定义了 EC2-Classic 安全组的入口规则。这些规则为 SSH（端口 22）和 RDC（端口 3389）授予对特定 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 TCP 端口 8081 从指定的源安全组（sg-1a2b3c4d）到指定的安全组（sg-12345678）的访问权限。

```
$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 5.5.5.5/32 添加到 TCP 端口 22 流量的安全组 sg-1234abcd 的入口规则中，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 (V 4) [AuthorizeSecurityGroupIngress](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例定义了 EC2-VPC 安全组的入口规则。这些规则为 SSH (端口 22) 和 RDC (端口 3389) 授予对特定 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：此示例定义了 EC2-Classic 安全组的入口规则。这些规则为 SSH（端口 22）和 RDC（端口 3389）授予对特定 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 TCP 端口 8081 从指定的源安全组（sg-1a2b3c4d）到指定的安全组（sg-12345678）的访问权限。

```
$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 5.5.5.5/32 添加到 TCP 端口 22 流量的安全组 sg-1234abcd 的入口规则中，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 \(V 5\) AuthorizeSecurityGroupIngress](#) 中的。

Python

适用于 Python 的 SDK (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。

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

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
    None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
        security group ID.
```

```

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                                access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
high-level identifier
                                that represents the security group.
    """
    self.ec2_client = ec2_client
    self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def authorize_ingress(self, ssh_ingress_ip: str) -> Optional[Dict[str, Any]]:
        """
        Adds a rule to the security group to allow access to SSH.

        :param ssh_ingress_ip: The IP address that is granted inbound access to
connect
                                to port 22 over TCP, used for SSH.
        :return: The response to the authorization request. The 'Return' field of
the
                                response indicates whether the request succeeded or failed, or
None if no security group is set.
        :raise Handles AWS SDK service-level ClientError, with special handling
for ResourceAlreadyExists
        """
        if self.security_group is None:
            logger.info("No security group to update.")
            return None

        try:
            ip_permissions = [
                {

```



```

        # SSH ingress open to only the specified IP address.
        "IpProtocol": "tcp",
        "FromPort": 22,
        "ToPort": 22,
        "IpRanges": [{"CidrIp": f"{ssh_ingress_ip}/32"}],
    }
]
response = self.ec2_client.authorize_security_group_ingress(
    GroupId=self.security_group, IpPermissions=ip_permissions
)
except ClientError as err:
    if err.response["Error"]["Code"] == "InvalidPermission.Duplicate":
        logger.error(
            f"The SSH ingress rule for IP {ssh_ingress_ip} already
exists"

            f"in security group '{self.security_group}'."
        )
        raise
    else:
        return response

```

- 有关 API 的详细信息，请参阅适用[AuthorizeSecurityGroupIngress](#)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](#) 中查找完整示例，了解如何进行设置和运行。

```

/// Add an ingress rule to a security group explicitly allowing IPv4 address
/// as {ip}/32 over TCP port 22.
pub async fn authorize_security_group_ssh_ingress(
    &self,
    group_id: &str,

```

```

        ingress_ips: Vec<Ipv4Addr>,
    ) -> Result<(), EC2Error> {
        tracing::info!("Authorizing ingress for security group {group_id}");
        self.client
            .authorize_security_group_ingress()
            .group_id(group_id)
            .set_ip_permissions(Some(
                ingress_ips
                    .into_iter()
                    .map(|ip| {
                        IpPermission::builder()
                            .ip_protocol("tcp")
                            .from_port(22)
                            .to_port(22)
                            .ip_ranges(IpRange::builder().cidr_ip(format!(
                                "{ip}/32"))).build())
                            .build()
                    })
                .collect(),
            ))
            .send()
            .await?;
        Ok(())
    }

```

- 有关 API 的详细信息，请参阅适用[AuthorizeSecurityGroupIngress](#)于 Rust 的AWS SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

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

```

" Create IP permissions for SSH access (port 22)
" iv_cidr_ip = '192.0.2.0/24'

```

```

DATA lt_ip_permissions TYPE /aws1/cl_ec2ippermission=>tt_ippermissionlist.
DATA(lo_ip_permission) = NEW /aws1/cl_ec2ippermission(
    iv_ipprotocol = 'tcp'
    iv_fromport = 22
    iv_toport = 22
    it_ipranges = VALUE /aws1/cl_ec2iprange=>tt_iprangelist(
        ( NEW /aws1/cl_ec2iprange( iv_cidrip = iv_cidr_ip ) )
    )
).
APPEND lo_ip_permission TO lt_ip_permissions.

TRY.
    oo_result = lo_ec2->authsecuritygroupingress(          " oo_result is
returned for testing purposes. "
    iv_groupid = iv_group_id
    it_ippermissions = lt_ip_permissions ).
    MESSAGE 'Authorized ingress rule for security group.' 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[AuthorizeSecurityGroupIngress](#)于 S AP 的AWS SDK ABAP API 参考。

Swift

适用于 Swift 的 SDK

Note

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

```

import AWSEC2

/// Authorize ingress of connections for the security group.
///

```

```

/// - Parameters:
///   - groupId: The group ID of the security group to authorize access for.
///   - ipAddress: The IP address of the device to grant access to.
///
/// - Returns: `true` if access is successfully granted; otherwise `false`.
func authorizeSecurityGroupIngress(groupId: String, ipAddress: String) async
-> Bool {
    let ipRange = EC2ClientTypes.IpRange(cidrIp: "\(ipAddress)/0")
    let httpPermission = EC2ClientTypes.IpPermission(
        fromPort: 80,
        ipProtocol: "tcp",
        ipRanges: [ipRange],
        toPort: 80
    )

    let sshPermission = EC2ClientTypes.IpPermission(
        fromPort: 22,
        ipProtocol: "tcp",
        ipRanges: [ipRange],
        toPort: 22
    )

    do {
        _ = try await ec2Client.authorizeSecurityGroupIngress(
            input: AuthorizeSecurityGroupIngressInput(
                groupId: groupId,
                ipPermissions: [httpPermission, sshPermission]
            )
        )

        return true
    } catch {
        print("*** Error authorizing ingress for the security group:
\\(error.localizedDescription)")
        return false
    }
}

```

- 有关 API 的详细信息，请参阅适用于 Swift 的 AWS SDK API 参考 [AuthorizeSecurityGroupIngress](#) 中。

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 SDK 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。

将 **CancelCapacityReservation** 与 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 示例将取消指定的容量预留。

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

输出：

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

有关更多信息，请参阅 Amazon EC2 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

适用于 PowerShell V4 的工具

示例 1：此示例取消容量预留 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 \(V 4\) `CancelCapacityReservation`](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例取消容量预留 `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 \(V 5\) `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 SDK 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。

将 `CancelImportTask` 与 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` 示例将取消指定导入映像任务。

```
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

适用于 PowerShell V4 的工具

示例 1：此示例取消指定的导入任务（快照或映像导入）。如果需要，可以使用 **CancelReason** 参数提供原因。

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

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CancelImportTask](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例取消指定的导入任务（快照或映像导入）。如果需要，可以使用 **CancelReason** 参数提供原因。

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

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) [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 SDK 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。

将 **CancelSpotFleetRequests** 与 CLI 配合使用

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

CLI

AWS CLI

示例 1：取消竞价型实例集请求并终止关联的实例

以下 `cancel-spot-fleet-requests` 示例将取消竞价型实例集请求并终止关联的按需型实例和竞价型实例。

```
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": []  
}
```

示例 2：取消竞价型实例集请求而不终止关联的实例

以下 `cancel-spot-fleet-requests` 示例将取消竞价型实例集请求而不终止关联的按需型实例和竞价型实例。

```
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 Amazon EC2 用户指南中的[取消 Spot 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

适用于 PowerShell V4 的工具

示例 1：此示例取消指定的竞价型实例集请求并终止关联的竞价型实例。

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

示例 2：此示例取消指定的竞价型实例集请求，但不终止关联的竞价型实例。

```
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 (V 4)
[CancelSpotFleetRequests](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例取消指定的竞价型实例集请求并终止关联的竞价型实例。

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

示例 2：此示例取消指定的竞价型实例集请求，但不终止关联的竞价型实例。

```
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 \(V 5\) `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 SDK 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。

将 `CancelSpotInstanceRequests` 与 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

此示例命令将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

适用于 PowerShell V4 的工具

示例 1：此示例取消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 \(V 4\) `CancelSpotInstanceRequests`](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例取消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 \(V 5\) `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 SDK 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。

将 `ConfirmProductInstance` 与 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

适用于 PowerShell V4 的工具

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 (V 4) [ConfirmProductInstance](#)中的。

适用于 PowerShell V5 的工具

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 (V 5) [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 SDK 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。

将 CopyImage 与 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 复制到其他区域

以下 `copy-image` 示例命令将指定的 AMI 从 `us-west-2` 区域复制到 `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 复制到其他区域并加密备份快照

以下 `copy-image` 命令将指定的 AMI 从 `us-west-2` 区域复制到当前区域，并使用指定的 KMS 密钥加密备份快照。

```
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

适用于 PowerShell V4 的工具

示例 1：此示例将“欧洲地区（爱尔兰）”区域中的指定 AMI 复制到“美国西部（俄勒冈州）”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 (V 4) [CopyImage](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例将“欧洲地区（爱尔兰）”区域中的指定 AMI 复制到“美国西部（俄勒冈州）”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 (V 5) [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 SDK 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。

将 CopySnapshot 与 CLI 配合使用

以下代码示例演示如何使用 CopySnapshot。

CLI

AWS CLI

示例 1：将快照复制到其他区域

以下 `copy-snapshot` 示例命令将指定的快照从 `us-west-2` 区域复制到 `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"
}
```

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 `us-west-2` 区域复制到当前区域，并使用指定的 KMS 密钥加密新快照。

```
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 EBS User Guide》中的 [Copy an Amazon EBS snapshot](#)。

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将指定的快照从欧洲地区（爱尔兰）区域复制到美国西部（俄勒冈州）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 (V 4) [CopySnapshot](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例将指定的快照从欧洲地区（爱尔兰）区域复制到美国西部（俄勒冈州）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 \(V 5\)](#) [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 SDK 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。

将 **CreateCapacityReservation** 与 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，在您手动取消容量预留之前，它将保持活动状态。

```
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：创建只接受目标实例启动的容量预留

以下 `create-capacity-reservation` 示例将创建一个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 Amazon EC2 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

适用于 PowerShell V4 的工具

示例 1：此示例使用指定的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 (V 4) [CreateCapacityReservation](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例使用指定的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 \(V 5\) 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 SDK 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。

将 `CreateCustomerGateway` 与 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

适用于 PowerShell V4 的工具

示例 1：此示例创建指定的客户网关。

```
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 (V 4) [CreateCustomerGateway](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例创建指定的客户网关。

```
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 \(V 5\) 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 SDK 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。

将 **CreateDhcpOptions** 与 CLI 配合使用

以下代码示例演示如何使用 CreateDhcpOptions。

CLI

AWS CLI

创建 DHCP 选项集

以下 create-dhcp-options 示例创建 DHCP 选项集，该选项集用于指定域名、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

适用于 PowerShell V4 的工具

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 \(V 4\)](#) [CreateDhcpOptions](#) 中的。

适用于 PowerShell V5 的工具

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 \(V 5\)](#) [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 SDK 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。

将 **CreateFlowLogs** 与 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 示例将创建流日志，该日志将捕获指定网络接口的所有已拒绝流量。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 示例将创建流日志，以捕获指定 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-0011223344556677 \
  --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 示例将创建流日志，以捕获指定 VPC 的所有流量并将流日志传输到 Amazon S3 存储桶。--max-aggregation-interval 参数指定的最大聚合间隔为 60 秒 (1 分钟)。

```
aws ec2 create-flow-logs \
  --resource-type VPC \
  --resource-ids vpc-0011223344556677 \
  --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

适用于 PowerShell V4 的工具

示例 1：此示例使用“管理员”角色的权限为所有“拒绝”EC2 流量创建子网 subnet-1d234567 到 cloud-watch-log名为“subnet1-log”的流日志

```
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//VUK15EU6cF7gQL0xcqNefvjeTGw=	{f1-012fc34eed5678c9d}	{}

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [CreateFlowLogs](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例使用“管理员”角色的权限为所有“拒绝”EC2 流量创建子网 subnet-1d234567 到 cloud-watch-log名为“subnet1-log”的流日志

```
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//VUK15EU6cF7gQL0xcqNefvjeTGw=	{f1-012fc34eed5678c9d}	{}

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) [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 SDK 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。

将 **CreateImage** 与 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 示例从指定的实例创建 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 AMI 并设置 --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 AMI，并使用相同的标签 cost-center=cc123 标记 AMI 和快照。

```
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"
}
```

有关在创建资源时为资源[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

适用于 PowerShell V4 的工具

示例 1：此示例从指定的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：此示例从指定的实例创建具有指定名称和描述的 AMI。Amazon 在不关闭和重启实例的情况下 EC2 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 Amazon EBS 卷。第三卷是一个实例存储卷。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 (V 4) [CreateImage](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例从指定的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：此示例从指定的实例创建具有指定名称和描述的 AMI。Amazon 在不关闭和重启实例的情况下 EC2 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 Amazon EBS 卷。第三卷是一个实例存储卷。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 (V 5) [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 SDK 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。

将 `CreateInstanceExportTask` 与 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

此示例命令将创建一个任务，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

适用于 PowerShell V4 的工具

示例 1：此示例将已停止的实例 **i-0800b00a00EXAMPLE** 作为虚拟硬盘（VHD）导出到 S3 存储桶 **testbucket-export-instances-2019**。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，请复制此命令的结果中的 **ExportTaskId** 值，然后运行 **Get-EC2ExportTask -ExportTaskId export_task_ID_from_results**。

```
New-EC2InstanceExportTask -InstanceId i-0800b00a00EXAMPLE -
ExportToS3Task_DiskImageFormat VHD -ExportToS3Task_S3Bucket "amzn-s3-demo-bucket"
-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 (V 4) [CreateInstanceExportTask](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例将已停止的实例 **i-0800b00a00EXAMPLE** 作为虚拟硬盘（VHD）导出到 S3 存储桶 **testbucket-export-instances-2019**。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，请复制此命令的结果中的 **ExportTaskId** 值，然后运行 **Get-EC2ExportTask -ExportTaskId export_task_ID_from_results**。

```
New-EC2InstanceExportTask -InstanceId i-0800b00a00EXAMPLE -
ExportToS3Task_DiskImageFormat VHD -ExportToS3Task_S3Bucket "amzn-s3-demo-bucket"
-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 \(V 5\) 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 SDK 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。

将 **CreateInternetGateway** 与 CLI 配合使用

以下代码示例演示如何使用 CreateInternetGateway。

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：

- [创建具有私有子网和 NAT 网关的 VPC](#)
- [开始使用 Amazon VPC](#)

CLI

AWS CLI

创建互联网网关

以下 create-internet-gateway 示例将创建带有标签 Name=my-igw 的互联网网关。

```
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

适用于 PowerShell V4 的工具

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 (V 4) [CreateInternetGateway](#)中的。

适用于 PowerShell V5 的工具

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 \(V 5\)](#) [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 SDK 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。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)
- [开始使用 Amazon VPC](#)

.NET

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。

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

```
{
    var request = new CreateKeyPairRequest { KeyName = keyPairName, };

    var response = await _amazonEC2.CreateKeyPairAsync(request);

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

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

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

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

```

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

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

        return pemFileName;
    }

```

- 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](#) 中查找完整示例，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 getopt command in a function.

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

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

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

```

```

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

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

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

return 0
}

```

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](#) 中查找完整示例，了解如何进行设置和运行。

```
//! Create an Amazon Elastic Compute Cloud (Amazon EC2) instance key pair.
/!*
\param keyPairName: A name for a key pair.
```

```

    \param keyFilePath: File path where the credentials are stored. Ignored if it
    is an empty string;
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
bool AwsDoc::EC2::createKeyPair(const Aws::String &keyPairName, const Aws::String
&keyFilePath,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::CreateKeyPairRequest request;
    request.SetKeyName(keyPairName);

    Aws::EC2::Model::CreateKeyPairOutcome outcome =
ec2Client.CreateKeyPair(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to create key pair - " << keyPairName << ". " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully created key pair named " <<
            keyPairName << std::endl;
        if (!keyFilePath.empty()) {
            std::ofstream keyFile(keyFilePath.c_str());
            keyFile << outcome.GetResult().GetKeyMaterial();
            keyFile.close();
            std::cout << "Keys written to the file " <<
                keyFilePath << std::endl;
        }
    }

    return outcome.IsSuccess();
}

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

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

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

```

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

    return responseFuture;
}

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```

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

/**
 * Creates an ED25519 or 2048-bit RSA key pair with the specified name and in the
 * specified PEM or PPK format.
 * Amazon EC2 stores the public key and displays the private key for you to save
 * to a file.
 * @param {{ keyName: string }} options
 */
export const main = async ({ keyName }) => {
    const client = new EC2Client({});
    const command = new CreateKeyPairCommand({

```

```
        KeyName: keyName,
    });

    try {
        const { KeyMaterial, KeyName } = await client.send(command);
        console.log(KeyName);
        console.log(KeyMaterial);
    } catch (caught) {
        if (caught instanceof Error && caught.name === "InvalidKeyPair.Duplicate") {
            console.warn(`${caught.message}. Try another key name.`);
        } else {
            throw caught;
        }
    }
};
```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```
suspend fun createEC2KeyPair(keyNameVal: String) {
    val request =
        CreateKeyPairRequest {
            keyName = keyNameVal
        }

    Ec2Client.fromEnvironment { 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 Kotlin 的 AWS SDK API 参考。

PowerShell

适用于 PowerShell V4 的工具

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 私有密钥。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、显示和删除亚马逊 EC2 密钥对 (<https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-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 (V 4) [CreateKeyPair](#)中的。

适用于 PowerShell V5 的工具

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 私有密钥。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、显示和删除亚马逊 EC2 密钥对 (<https://docs.aws.amazon.com/cli/latest/userguide/cli-services-ec2-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 (V 5) [CreateKeyPair](#)中的。

Python

适用于 Python 的 SDK (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。

```

class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
        directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is
stored.
                           This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                           This is a high-level object that wraps key pair actions.
Optional.
        """
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2
client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())

```

```

def create(self, key_name: str) -> dict:
    """
    Creates a key pair that can be used to securely connect to an EC2
    instance.
    The returned key pair contains private key information that cannot be
    retrieved
    again. The private key data is stored as a .pem file.

    :param key_name: The name of the key pair to create.
    :return: A dictionary representing the Boto3 KeyPair object that
    represents the newly created key pair.
    :raises ClientError: If there is an error in creating the key pair, for
    example, if a key pair with the same name already exists.
    """
    try:
        response = self.ec2_client.create_key_pair(KeyName=key_name)
        self.key_pair = response
        self.key_file_path = os.path.join(
            self.key_file_dir.name, f"{self.key_pair['KeyName']}.pem"
        )
        with open(self.key_file_path, "w") as key_file:
            key_file.write(self.key_pair["KeyMaterial"])
    except ClientError as err:
        if err.response["Error"]["Code"] == "InvalidKeyPair.Duplicate":
            logger.error(
                f"A key pair called {key_name} already exists. "
                "Please choose a different name for your key pair "
                "or delete the existing key pair before creating."
            )
            raise
    else:
        return self.key_pair

```

- 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](#) 中查找完整示例，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}'."
  true
rescue Aws::EC2::Errors::InvalidKeyPairDuplicate
  puts "Error creating key pair: a key pair named '#{key_pair_name}' " \
    'already exists.'
  false
rescue StandardError => e
  puts "Error creating key pair or saving private key file: #{e.message}"
```

```

    false
  end

  # Displays information about available key pairs in
  # Amazon Elastic Compute Cloud (Amazon EC2).
  #
  # @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
  # @example
  #   describe_key_pairs(Aws::EC2::Client.new(region: 'us-west-2'))
  def describe_key_pairs(ec2_client)
    result = ec2_client.describe_key_pairs
    if result.key_pairs.count.zero?
      puts 'No key pairs found.'
    else
      puts 'Key pair names:'
      result.key_pairs.each do |key_pair|
        puts key_pair.key_name
      end
    end
  end
rescue StandardError => e
  puts "Error getting information about key pairs: #{e.message}"
end

# Deletes a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - The key pair to delete.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name of the key pair to delete.
# @return [Boolean] true if the key pair was deleted; otherwise, false.
# @example
#   exit 1 unless key_pair_deleted?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_deleted?(ec2_client, key_pair_name)
  ec2_client.delete_key_pair(key_name: key_pair_name)
  true
rescue StandardError => e
  puts "Error deleting key pair: #{e.message}"
  false
end

```

```
# Example usage:
def run_me
  key_pair_name = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-key-pairs.rb KEY_PAIR_NAME REGION'
    puts 'Example: ruby ec2-ruby-example-key-pairs.rb my-key-pair us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    key_pair_name = 'my-key-pair'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    key_pair_name = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts 'Displaying existing key pair names before creating this key pair...'
  describe_key_pairs(ec2_client)

  puts '-' * 10
  puts 'Creating key pair...'
  unless key_pair_created?(ec2_client, key_pair_name)
    puts 'Stopping program.'
    exit 1
  end

  puts '-' * 10
  puts 'Displaying existing key pair names after creating this key pair...'
  describe_key_pairs(ec2_client)

  puts '-' * 10
  puts 'Deleting key pair...'
  unless key_pair_deleted?(ec2_client, key_pair_name)
    puts 'Stopping program. You must delete the key pair yourself.'
    exit 1
  end
  puts 'Key pair deleted.'
```

```
puts '-' * 10
puts 'Now that the key pair is deleted, ' \
    'also deleting the related private key pair file...'
filename = File.join(Dir.home, "#{key_pair_name}.pem")
File.delete(filename)
if File.exist?(filename)
  puts "Could not delete file at '#{filename}'. You must delete it yourself."
else
  puts 'File deleted.'
end

puts '-' * 10
puts 'Displaying existing key pair names after deleting this key pair...'
describe_key_pairs(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考[CreateKeyPair](#)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](#) 中查找完整示例，了解如何进行设置和运行。

Rust 实现，它调用 EC2 客户端的 `create_key_pair` 并提取返回的材料。

```
pub async fn create_key_pair(&self, name: String) -> Result<(KeyPairInfo,
String), EC2Error> {
    tracing::info!("Creating key pair {name}");
    let output = self.client.create_key_pair().key_name(name).send().await?;
    let info = KeyPairInfo::builder()
        .set_key_name(output.key_name)
        .set_key_fingerprint(output.key_fingerprint)
        .set_key_pair_id(output.key_pair_id)
```

```

        .build();
    let material = output
        .key_material
        .ok_or_else(|| EC2Error::new("Create Key Pair has no key
material"))?;
    Ok((info, material))
}

```

一个函数，它调用 `create_key impl` 并安全地保存 PEM 私有密钥。

```

    /// Creates a key pair that can be used to securely connect to an EC2
    instance.
    /// The returned key pair contains private key information that cannot be
    retrieved
    /// again. The private key data is stored as a .pem file.
    ///
    /// :param key_name: The name of the key pair to create.
    pub async fn create(
        &mut self,
        ec2: &EC2,
        util: &Util,
        key_name: String,
    ) -> Result<KeyPairInfo, EC2Error> {
        let (key_pair, material) =
ec2.create_key_pair(key_name.clone()).await.map_err(|e| {
            self.key_pair =
KeyPairInfo::builder().key_name(key_name.clone()).build();
            e.add_message(format!("Couldn't create key {key_name}"))
        })?;

        let path = self.key_file_dir.join(format!("{key_name}.pem"));

        // Save the key_pair information immediately, so it can get cleaned up if
        write_secure fails.
        self.key_file_path = Some(path.clone());
        self.key_pair = key_pair.clone();

        util.write_secure(&key_name, &path, material)?;

        Ok(key_pair)
    }

```

- 有关 API 的详细信息，请参阅适用[CreateKeyPair](#)于 Rust 的 AWS SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，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。

Swift

适用于 Swift 的 SDK

Note

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

```
import AWSEC2  
  
/// Create a new RSA key pair and save the private key to a randomly-named  
/// file in the temporary directory.
```

```

///
/// - Parameter name: The name of the key pair to create.
///
/// - Returns: The URL of the newly created `.pem` file or `nil` if unable
///   to create the key pair.
func createKeyPair(name: String) async -> URL? {
    do {
        let output = try await ec2Client.createKeyPair(
            input: CreateKeyPairInput(
                keyName: name
            )
        )

        guard let keyMaterial = output.keyMaterial else {
            return nil
        }

        // Build the URL of the temporary private key file.

        let fileURL = URL.temporaryDirectory
            .appendingPathComponent(name)
            .appendingPathExtension("pem")

        do {
            try keyMaterial.write(to: fileURL, atomically: true, encoding:
String.Encoding.utf8)
            return fileURL
        } catch {
            print("*** Failed to write the private key.")
            return nil
        }
    } catch {
        print("*** Unable to create the key pair.")
        return nil
    }
}

```

- 有关 API 的详细信息，请参阅适用于 Swift 的 AWS SDK API 参考 [CreateKeyPair](#) 中。

有关 AWS SDK 开发者指南和代码示例的完整列表，请参阅 [使用 AWS SDK 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。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](#)
- [创建具有私有子网和 NAT 网关的 VPC](#)

.NET

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。

```
/// <summary>
/// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
/// The launch template specifies a Bash script in its user data field that
runs after
/// the instance is started. This script installs the Python packages and
starts a Python
/// web server on the instance.
/// </summary>
/// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
/// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
/// <returns>The template object.</returns>
public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
{
    try
    {
        await CreateKeyPair(_keyPairName);
```



```

        await CreateInstanceProfileWithName(_instancePolicyName,
            _instanceRoleName,
            _instanceProfileName, instancePolicyPath);

        var startServerText = await File.ReadAllTextAsync(startupScriptPath);
        var plainTextBytes =
            System.Text.Encoding.UTF8.GetBytes(startServerText);

        var amiLatest = await _amazonSsm.GetParameterAsync(
            new GetParameterRequest() { Name = _amiParam });
        var amiId = amiLatest.Parameter.Value;
        var launchTemplateResponse = await
            _amazonEc2.CreateLaunchTemplateAsync(
                new CreateLaunchTemplateRequest()
                {
                    LaunchTemplateName = _launchTemplateName,
                    LaunchTemplateData = new RequestLaunchTemplateData()
                    {
                        InstanceType = _instanceType,
                        ImageId = amiId,
                        IamInstanceProfile =
                            new
                                LaunchTemplateIamInstanceProfileSpecificationRequest()
                                {
                                    Name = _instanceProfileName
                                },
                        KeyName = _keyPairName,
                        UserData = System.Convert.ToBase64String(plainTextBytes)
                    }
                });
        return launchTemplateResponse.LaunchTemplate;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode ==
            "InvalidLaunchTemplateName.AlreadyExistsException")
        {
            _logger.LogError($"Could not create the template, the name
                {_launchTemplateName} already exists. " +
                    $"Please try again with a unique name.");
        }

        throw;
    }

```

```

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

```

- 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示例创建了一个启动模板，该模板指定了启动实例的子网，为该实例分配了公有 IP IPv6 地址和地址，并为该实例创建了标签。

```

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 A EC2 uto 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 Auto Scaling 用户指南中的为 Auto EC2 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](#) 中查找完整示例，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

适用于 Python 的 SDK (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。

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 AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

```

```

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
        Scaling. The
        launch template specifies a Bash script in its user data field that runs
        after
        the instance is started. This script installs Python packages and starts
        a

```

```

        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.

        :param instance_policy_file: The path to a file that defines a
permissions policy
                                           to create and attach to the instance
profile.

        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]

```



```
        log.info(
            f"Created launch template {self.launch_template_name} for AMI {ami_id} on {self.inst_type}."
        )
    except ClientError as err:
        log.error(f"Failed to create launch template {self.launch_template_name}.")
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
            log.info(
                f"Launch template {self.launch_template_name} already exists, nothing to do."
            )
        log.error(f"Full error:\n\t{err}")
    return template
```

- 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 SDK 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。

将 **CreateNetworkAcl** 与 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

此示例将为指定的 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

适用于 PowerShell V4 的工具

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 \(V 4\) CreateNetworkAcl](#) 中的。

适用于 PowerShell V5 的工具

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 \(V 5\) 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 SDK 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。

将 `CreateNetworkAclEntry` 与 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

此示例将为指定的网络 ACL 创建一个条目。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，该规则允许来自 TCP 端口 80 (HTTP) 上的任何 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

适用于 PowerShell V4 的工具

示例 1：此示例为指定的网络 ACL 创建一个条目。该规则可让来自 UDP 端口 53 (DNS) 上任何地址 (0.0.0.0/0) 的入站流量进入任何关联的子网。

```
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 (V 4) [CreateNetworkAclEntry](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例为指定的网络 ACL 创建一个条目。该规则可让来自 UDP 端口 53 (DNS) 上任何地址 (0.0.0.0/0) 的入站流量进入任何关联的子网。

```
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 \(V 5\) `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 SDK 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。

将 `CreateNetworkInterface` 与 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：使用 IPv4 地址和地址创建网络接口 IPv6

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

```

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` 示例将创建网络接口并配置空闲连接跟踪超时。

```

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-02e57dbcf0331c1b"
        }
    ],
    "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 Elastic Fabric Adapter

以下 create-network-interface 示例将创建 EFA。

```

aws ec2 create-network-interface \
  --interface-type efa \
  --subnet-id subnet-00a24d0d67acf6333 \
  --description "my efa" \
  --groups sg-02e57dbcf0331c1b

```


输出：

```
{
  "NetworkInterface": {
    "AvailabilityZone": "us-west-2a",
    "Description": "my efa",
    "Groups": [
      {
        "GroupName": "my-efa-sg",
        "GroupId": "sg-02e57dbcf0331c1b"
      }
    ],
    "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

适用于 PowerShell V4 的工具

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 \(V 4\) CreateNetworkInterface](#) 中的。

适用于 PowerShell V5 的工具

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 \(V 5\) `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 SDK 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。

将 `CreatePlacementGroup` 与 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

此示例命令将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

此示例命令将创建一个名为 `HDFS-Group-A` 的分区置放群组，其中包含 5 个分区。

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

适用于 PowerShell V4 的工具

示例 1：此示例命令使用指定的名称创建置放群组。

```
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 (V 4) [CreatePlacementGroup](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例命令使用指定的名称创建置放群组。

```
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 (V 5) [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 SDK 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。

将 **CreateRoute** 与 CLI 配合使用

以下代码示例演示如何使用 CreateRoute。

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：

- [创建具有私有子网和 NAT 网关的 VPC](#)
- [开始使用 Amazon VPC](#)
- [中转网关入门](#)

CLI

AWS CLI

创建路由

此示例将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 Internet 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。此路由可将流量定向到 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条匹配所有 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

适用于 PowerShell V4 的工具

示例 1：此示例为指定的路由表创建指定的路由。该路由匹配所有流量，并将其路由到指定的互联网网关。

```
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 (V 4) [CreateRoute](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例为指定的路由表创建指定的路由。该路由匹配所有流量，并将其路由到指定的互联网网关。

```
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 (V 5) [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 SDK 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。

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：

- [创建具有私有子网和 NAT 网关的 VPC](#)
- [开始使用 Amazon VPC](#)

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

适用于 PowerShell V4 的工具

示例 1：此示例为指定的 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 \(V 4\) CreateRouteTable](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例为指定的 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 \(V 5\) 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](#) 中查找完整示例，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
      }
    ]
  )
end

```

```

    ]
  )
  puts 'Added tags to route table.'
  route_table.create_route(
    destination_cidr_block: destination_cidr_block,
    gateway_id: gateway_id
  )
  puts 'Created route with destination CIDR block ' \
    "'#{destination_cidr_block}' and associated with gateway " \
    "with ID '#{gateway_id}'."
  route_table.associate_with_subnet(subnet_id: subnet_id)
  puts "Associated route table with subnet with ID '#{subnet_id}'."
  true
rescue StandardError => e
  puts "Error creating or associating route table: #{e.message}"
  puts 'If the route table was created but not associated, you should ' \
    'clean up by deleting the route table.'
  false
end

# Example usage:
def run_me
  vpc_id = ''
  subnet_id = ''
  gateway_id = ''
  destination_cidr_block = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage: ruby ec2-ruby-example-create-route-table.rb ' \
      'VPC_ID SUBNET_ID GATEWAY_ID DESTINATION_CIDR_BLOCK ' \
      'TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-route-table.rb ' \
      'vpc-0b6f769731EXAMPLE subnet-03d9303b57EXAMPLE igw-06ca90c011EXAMPLE ' \
      "'0.0.0.0/0' my-key my-value us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = 'vpc-0b6f769731EXAMPLE'
    subnet_id = 'subnet-03d9303b57EXAMPLE'
    gateway_id = 'igw-06ca90c011EXAMPLE'
  end
end

```

```
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 SDK 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。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)
- [创建具有私有子网和 NAT 网关的 VPC](#)
- [开始使用 Amazon VPC](#)

.NET

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。

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

        // Wait until the security group exists.
        int retries = 5;
        while (retries-- > 0)
```

```
        {
            var groups = await DescribeSecurityGroups(response.GroupId);
            if (groups.Any())
            {
                return response.GroupId;
            }

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

- 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](#) 中查找完整示例，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](#) 中查找完整示例，了解如何进行设置和运行。

```

//! Create a security group.
/*!
    \param groupName: A security group name.
    \param description: A description.
    \param vpcID: A virtual private cloud (VPC) ID.
    \param[out] groupIDResult: A string to receive the group ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::createSecurityGroup(const Aws::String &groupName,
                                     const Aws::String &description,
                                     const Aws::String &vpcID,
                                     Aws::String &groupIDResult,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::CreateSecurityGroupRequest request;

    request.SetGroupName(groupName);
    request.SetDescription(description);
    request.SetVpcId(vpcID);

    const Aws::EC2::Model::CreateSecurityGroupOutcome outcome =

```

```
        ec2Client.CreateSecurityGroup(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to create security group:" <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    std::cout << "Successfully created security group named " << groupName <<
        std::endl;

    groupIDResult = outcome.GetResult().GetGroupId();

    return true;
}
```

- 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-Classical 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](#) 中查找完整示例，了解如何进行设置和运行。

```
/**
 * Creates a new security group asynchronously with the specified group name,
 * description, and VPC ID. It also
 *   * authorizes inbound traffic on ports 80 and 22 from the specified IP
 * address.
 *
 * @param groupName    the name of the security group to create
 * @param groupDesc    the description of the security group
 * @param vpcId        the ID of the VPC in which to create the security
 * group
 * @param myIpAddress  the IP address from which to allow inbound traffic
 * (e.g., "192.168.1.1/0" to allow traffic from
 *   *                  any IP address in the 192.168.1.0/24 subnet)
 * @return a CompletableFuture that, when completed, returns the ID of the
 * created security group
```

```

    * @throws RuntimeException if there was a failure creating the security
    group or authorizing the inbound traffic
    */
    public CompletableFuture<String> createSecurityGroupAsync(String groupName,
String groupDesc, String vpcId, String myIpAddress) {
        CreateSecurityGroupRequest createRequest =
CreateSecurityGroupRequest.builder()
            .groupName(groupName)
            .description(groupDesc)
            .vpcId(vpcId)
            .build();

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

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

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

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

                return
getAsyncClient().authorizeSecurityGroupIngress(authRequest)
                    .thenApply(authResponse -> groupId);
            })
            .whenComplete((result, exception) -> {

```

```

        if (exception != null) {
            if (exception instanceof CompletionException &&
exception.getCause() instanceof Ec2Exception) {
                throw (Ec2Exception) exception.getCause();
            } else {
                throw new RuntimeException("Failed to create security
group: " + exception.getMessage(), exception);
            }
        }
    });
}

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```

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

/**
 * Creates a security group.
 * @param {{ groupName: string, description: string }} options
 */
export const main = async ({ groupName, description }) => {
    const client = new EC2Client({});
    const command = new CreateSecurityGroupCommand({
        // Up to 255 characters in length. Cannot start with sg-.
        GroupName: groupName,
        // Up to 255 characters in length.
        Description: description,
    });

    try {
        const { GroupId } = await client.send(command);
    }
}

```

```
    console.log(GroupId);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}.`);
    } else {
      throw caught;
    }
  }
};
```

- 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](#) 中查找完整示例，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.fromEnvironment { 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 Kotlin 的 AWS SDK API 参考。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例为指定的 VPC 创建一个安全组。

```
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 \(V 4\) CreateSecurityGroup](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例为指定的 VPC 创建一个安全组。

```
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 \(V 5\) CreateSecurityGroup](#) 中的。

Python

适用于 Python 的 SDK (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。

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

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
    None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
        security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
        high-level identifier
                           that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
        EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```

def create(self, group_name: str, group_description: str) -> str:
    """
    Creates a security group in the default virtual private cloud (VPC) of
    the current account.

    :param group_name: The name of the security group to create.
    :param group_description: The description of the security group to
    create.
    :return: The ID of the newly created security group.
    :raise: Handles AWS SDK service-level ClientError, with special handling
    for ResourceAlreadyExists
    """
    try:
        response = self.ec2_client.create_security_group(
            GroupName=group_name, Description=group_description
        )
        self.security_group = response["GroupId"]
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceAlreadyExists":
            logger.error(
                f"Security group '{group_name}' already exists. Please choose
                a different name."
            )
            raise
        else:
            return self.security_group

```

- 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](#) 中查找完整示例，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}'."
  security_group.group_id
rescue StandardError => e
  puts "Error creating security group: #{e.message}"
  'Error'
end

# Adds an inbound rule to an Amazon Elastic Compute Cloud (Amazon EC2)

```

```

# security group.
#
# Prerequisites:
#
# - The security group.
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @param security_group_id [String] The ID of the security group.
# @param ip_protocol [String] The network protocol for the inbound rule.
# @param from_port [String] The originating port for the inbound rule.
# @param to_port [String] The destination port for the inbound rule.
# @param cidr_ip_range [String] The CIDR IP range for the inbound rule.
# @return
# @example
#   exit 1 unless security_group_ingress_authorized?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'sg-030a858e078f1b9EX',
#     'tcp',
#     '80',
#     '80',
#     '0.0.0.0/0'
#   )
def security_group_ingress_authorized?(
  ec2_client, security_group_id, ip_protocol, from_port, to_port, cidr_ip_range
)
  ec2_client.authorize_security_group_ingress(
    group_id: security_group_id,
    ip_permissions: [
      {
        ip_protocol: ip_protocol,
        from_port: from_port,
        to_port: to_port,
        ip_ranges: [
          {
            cidr_ip: cidr_ip_range
          }
        ]
      }
    ]
  )
  puts "Added inbound rule to security group '#{security_group_id}' for protocol
" \
  "'#{ip_protocol}' from port '#{from_port}' to port '#{to_port}' " \
  "with CIDR IP range '#{cidr_ip_range}'."

```

```

    true
  rescue StandardError => e
    puts "Error adding inbound rule to security group: #{e.message}"
    false
  end

  # Refactored method to simplify complexity for describing security group
  permissions
  def format_port_information(perm)
    from_port_str = perm.from_port == '-1' || perm.from_port == -1 ? 'All' :
    perm.from_port.to_s
    to_port_str = perm.to_port == '-1' || perm.to_port == -1 ? 'All' :
    perm.to_port.to_s
    { from_port: from_port_str, to_port: to_port_str }
  end

  # Displays information about a security group's IP permissions set in
  # Amazon Elastic Compute Cloud (Amazon EC2).
  def describe_security_group_permissions(perm)
    ports = format_port_information(perm)

    print "  Protocol: #{perm.ip_protocol == '-1' ? 'All' : perm.ip_protocol}"
    print ", From: #{ports[:from_port]}, To: #{ports[:to_port]}"

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

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

  # Displays information about available security groups in
  # Amazon Elastic Compute Cloud (Amazon EC2).
  def describe_security_groups(ec2_client)
    response = ec2_client.describe_security_groups

    if response.security_groups.count.positive?
      response.security_groups.each do |sg|
        display_group_details(sg)
      end
    else
      puts 'No security groups found.'
    end
  end

```

```
rescue StandardError => e
  puts "Error getting information about security groups: #{e.message}"
end

# Helper method to display the details of security groups
def display_group_details(sg)
  puts '-' * (sg.group_name.length + 13)
  puts "Name:      #{sg.group_name}"
  puts "Description: #{sg.description}"
  puts "Group ID:    #{sg.group_id}"
  puts "Owner ID:    #{sg.owner_id}"
  puts "VPC ID:      #{sg.vpc_id}"

  display_group_tags(sg.tags) if sg.tags.count.positive?
  display_group_permissions(sg)
end

def display_group_tags(tags)
  puts 'Tags:'
  tags.each do |tag|
    puts "  Key: #{tag.key}, Value: #{tag.value}"
  end
end

def display_group_permissions(sg)
  if sg.ip_permissions.count.positive?
    puts 'Inbound rules:'
    sg.ip_permissions.each do |p|
      describe_security_group_permissions(p)
    end
  end

  return if sg.ip_permissions_egress.empty?

  puts 'Outbound rules:'
  sg.ip_permissions_egress.each do |p|
    describe_security_group_permissions(p)
  end
end

# Deletes an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
def security_group_deleted?(ec2_client, security_group_id)
  ec2_client.delete_security_group(group_id: security_group_id)
```

```

    puts "Deleted security group '#{security_group_id}'."
    true
  rescue StandardError => e
    puts "Error deleting security group: #{e.message}"
    false
  end

# Example usage with refactored run_me to reduce complexity
def run_me
  group_name, description, vpc_id, ip_protocol_http, from_port_http,
  to_port_http, \
  cidr_ip_range_http, ip_protocol_ssh, from_port_ssh, to_port_ssh, \
  cidr_ip_range_ssh, region = process_arguments
  ec2_client = Aws::EC2::Client.new(region: region)

  security_group_id = attempt_create_security_group(ec2_client, group_name,
  description, vpc_id)
  security_group_exists = security_group_id != 'Error'

  if security_group_exists
    add_inbound_rules(ec2_client, security_group_id, ip_protocol_http,
    from_port_http, to_port_http, cidr_ip_range_http)
    add_inbound_rules(ec2_client, security_group_id, ip_protocol_ssh,
    from_port_ssh, to_port_ssh, cidr_ip_range_ssh)
  end

  describe_security_groups(ec2_client)
  attempt_delete_security_group(ec2_client, security_group_id) if
  security_group_exists
end

def process_arguments
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    display_help
    exit 1
  elsif ARGV.count.zero?
    default_values
  else
    ARGV
  end
end

def attempt_create_security_group(ec2_client, group_name, description, vpc_id)
  puts 'Attempting to create security group...'

```

```

    security_group_id = create_security_group(ec2_client, group_name, description,
vpc_id)
    puts 'Could not create security group. Skipping this step.' if
security_group_id == 'Error'
    security_group_id
end

def add_inbound_rules(ec2_client, security_group_id, ip_protocol, from_port,
to_port, cidr_ip_range)
    puts 'Attempting to add inbound rules to security group...'
    return if security_group_ingress_authorized?(ec2_client, security_group_id,
ip_protocol, from_port, to_port,
cidr_ip_range)

    puts 'Could not add inbound rule to security group. Skipping this step.'
end

def attempt_delete_security_group(ec2_client, security_group_id)
    puts "\nAttempting to delete security group..."
    return if security_group_deleted?(ec2_client, security_group_id)

    puts 'Could not delete security group. You must delete it yourself.'
end

def display_help
    puts 'Usage:  ruby ec2-ruby-example-security-group.rb ' \
'GROUP_NAME DESCRIPTION VPC_ID IP_PROTOCOL_1 FROM_PORT_1 TO_PORT_1 ' \
'CIDR_IP_RANGE_1 IP_PROTOCOL_2 FROM_PORT_2 TO_PORT_2 ' \
'CIDR_IP_RANGE_2 REGION'
    puts 'Example: ruby ec2-ruby-example-security-group.rb ' \
"my-security-group 'This is my security group.' vpc-6713dfEX " \
"tcp 80 80 '0.0.0.0/0' tcp 22 22 '0.0.0.0/0' us-west-2"
end

def default_values
    [
        'my-security-group', 'This is my security group.', 'vpc-6713dfEX', 'tcp',
'80', '80',
        '0.0.0.0/0', 'tcp', '22', '22', '0.0.0.0/0', 'us-west-2'
    ]
end

run_me if $PROGRAM_NAME == __FILE__

```


- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考 [CreateSecurityGroup](#) 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](#) 中查找完整示例，了解如何进行设置和运行。

```
pub async fn create_security_group(
    &self,
    name: &str,
    description: &str,
) -> Result<SecurityGroup, EC2Error> {
    tracing::info!("Creating security group {name}");
    let create_output = self
        .client
        .create_security_group()
        .group_name(name)
        .description(description)
        .send()
        .await
        .map_err(EC2Error::from)?;

    let group_id = create_output
        .group_id
        .ok_or_else(|| EC2Error::new("Missing security group id after
creation"))?;

    let group = self
        .describe_security_group(&group_id)
        .await?
        .ok_or_else(|| {
            EC2Error::new(format!("Could not find security group with id
{group_id}"))
        })?;
```

```
tracing::info!("Created security group {name} as {group_id}");

Ok(group)
}
```

- 有关 API 的详细信息，请参阅适用[CreateSecurityGroup](#)于 Rust 的AWS SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

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

```
TRY.
    oo_result = lo_ec2->createsecuritygroup(
        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。

Swift

适用于 Swift 的 SDK

Note

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

```
import AWSEC2

/// Create a new security group.
///
/// - Parameters:
///   - groupName: The name of the group to create.
///   - groupDescription: A description of the new security group.
///
/// - Returns: The ID string of the new security group.
func createSecurityGroup(name groupName: String, description
groupDescription: String) async -> String? {
    do {
        let output = try await ec2Client.createSecurityGroup(
            input: CreateSecurityGroupInput(
                description: groupDescription,
                groupName: groupName
            )
        )

        return output.groupId
    } catch {
        print("**** Error creating the security group:
\\(error.localizedDescription)")
        return nil
    }
}
```

- 有关 API 的详细信息，请参阅适用于 Swift 的 AWS SDK API 参考 [CreateSecurityGroup](#) 中。

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 SDK 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。

将 `CreateSnapshot` 与 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

此示例命令将创建卷 ID 为 `vol-1234567890abcdef0` 的卷的快照，并提供简短描述以标识快照。

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"
}
```

创建具有标签的快照

此示例命令将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

适用于 PowerShell V4 的工具

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 \(V 4\) CreateSnapshot](#) 中的。

适用于 PowerShell V5 的工具

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 \(V 5\) 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 SDK 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。

将 `CreateSpotDatafeedSubscription` 与 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` 示例将创建竞价型实例数据源。

```
aws ec2 create-spot-datafeed-subscription \
  --bucket amzn-s3-demo-bucket \
  --prefix spot-data-feed
```

输出：

```
{
  "SpotDatafeedSubscription": {
    "Bucket": "amzn-s3-demo-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。此数据源的文件名采用以下格式。

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

有关更多信息，请参阅 Amazon EC2 用户指南中的[竞价型实例数据 Feed](#)。

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例创建竞价型实例数据源。

```
New-EC2SpotDatafeedSubscription -Bucket amzn-s3-demo-bucket -Prefix spotdata
```

输出：

```
Bucket   : amzn-s3-demo-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 \(V 4\) CreateSpotDatafeedSubscription](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例创建竞价型实例数据源。

```
New-EC2SpotDatafeedSubscription -Bucket amzn-s3-demo-bucket -Prefix spotdata
```

输出：

```
Bucket   : amzn-s3-demo-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 \(V 5\) 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 SDK 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。

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：

- [创建具有私有子网和 NAT 网关的 VPC](#)
- [开始使用 Amazon VPC](#)
- [中转网关入门](#)

CLI

AWS CLI

示例 1：创建仅包含 IPv4 CIDR 块的子网

以下create-subnet示例使用指定 IPv4 CIDR 块在指定 VPC 中创建子网。

```
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：创建同时包含 IPv4 和 IPv6 CIDR 块的子网

以下 `create-subnet` 示例使用指定的 IPv4 和 IPv6 CIDR 块在指定 VPC 中创建一个子网。

```

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：创建仅包含 IPv6 CIDR 块的子网

以下create-subnet示例使用指定 IPv6 CIDR 块在指定 VPC 中创建子网。

```

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[VPCs 和子网](#)。

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例使用指定的 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 (V 4) [CreateSubnet](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例使用指定的 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 (V 5) [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](#) 中查找完整示例，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_value [String] The value portion of the tag for the subnet.
# @return [Boolean] true if the subnet was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless subnet_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-6713dfEX',
#     '10.0.0.0/24',
#     'us-west-2a',
#     'my-key',
#     'my-value'
#   )
def subnet_created_and_tagged?(
  ec2_resource,
  vpc_id,
  cidr_block,
  availability_zone,
  tag_key,
  tag_value
)
  subnet = ec2_resource.create_subnet(
    vpc_id: vpc_id,
    cidr_block: cidr_block,
    availability_zone: availability_zone
  )
  subnet.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts "Subnet created with ID '#{subnet.id}' in VPC with ID '#{vpc_id}' " \
    "and CIDR block '#{cidr_block}' in availability zone " \
    "'#{availability_zone}' and tagged with key '#{tag_key}' and " \
    "value '#{tag_value}'."
  true
rescue StandardError => e

```

```

    puts "Error creating or tagging subnet: #{e.message}"
    false
end

# Example usage:
def run_me
  vpc_id = ''
  cidr_block = ''
  availability_zone = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-create-subnet.rb ' \
      'VPC_ID CIDR_BLOCK AVAILABILITY_ZONE TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-subnet.rb ' \
      'vpc-6713dfEX 10.0.0.0/24 us-west-2a my-key my-value us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = 'vpc-6713dfEX'
    cidr_block = '10.0.0.0/24'
    availability_zone = 'us-west-2a'
    tag_key = 'my-key'
    tag_value = 'my-value'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
    cidr_block = ARGV[1]
    availability_zone = ARGV[2]
    tag_key = ARGV[3]
    tag_value = ARGV[4]
    region = ARGV[5]
  end

  ec2_resource = Aws::EC2::Resource.new(region: region)

  if subnet_created_and_tagged?(
    ec2_resource,
    vpc_id,

```

```

        cidr_block,
        availability_zone,
        tag_key,
        tag_value
    )
    puts 'Subnet created and tagged.'
else
    puts 'Subnet not created or not tagged.'
end
end
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 SDK 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](#) 中查找完整示例，了解如何进行设置和运行。

```

//! Add or overwrite only the specified tags for the specified Amazon Elastic
    Compute Cloud (Amazon EC2) resource or resources.
/*!
    \param resources: The resources for the tags.
    \param tags: Vector of tags.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/

```



```
bool AwsDoc::EC2::createTags(const Aws::Vector<Aws::String> &resources,
                             const Aws::Vector<Aws::EC2::Model::Tag> &tags,
                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::CreateTagsRequest createTagsRequest;
    createTagsRequest.SetResources(resources);
    createTagsRequest.SetTags(tags);

    Aws::EC2::Model::CreateTagsOutcome outcome =
    ec2Client.CreateTags(createTagsRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created tags for resources" << std::endl;
    } else {
        std::cerr << "Failed to create tags for resources, " <<
        outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 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：将标签添加到资源

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
```

此命令不生成任何输出

示例 2：将标签添加到多个资源

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
```

此命令不生成任何输出

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《[亚马逊 EC2 用户指南](#)》中的[为您的亚马逊 EC2 资源添加标签](#)。

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例向指定的资源添加单个标签。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：此示例更新指定的标签，或将其添加到指定的资源。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 (V 4) [CreateTags](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例向指定的资源添加单个标签。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：此示例更新指定的标签，或将其添加到指定的资源。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 (V 5) [CreateTags](#) 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](#) 中查找完整示例，了解如何进行设置和运行。

此示例在创建实例后应用 Name 标签。

```
pub async fn create_instance<'a>(
    &self,
    image_id: &'a str,
    instance_type: InstanceType,
    key_pair: &'a KeyPairInfo,
    security_groups: Vec<&'a SecurityGroup>,
) -> Result<String, EC2Error> {
    let run_instances = self
        .client
        .run_instances()
        .image_id(image_id)
        .instance_type(instance_type)
        .key_name(
            key_pair
                .key_name()
                .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?,
        )
        .set_security_group_ids(Some(
            security_groups
```

```

        .iter()
        .filter_map(|sg| sg.group_id.clone())
        .collect(),
    ))
    .min_count(1)
    .max_count(1)
    .send()
    .await?;

if run_instances.instances().is_empty() {
    return Err(EC2Error::new("Failed to create instance"));
}

let instance_id = run_instances.instances()[0].instance_id().unwrap();
let response = self
    .client
    .create_tags()
    .resources(instance_id)
    .tags(
        Tag::builder()
            .key("Name")
            .value("From SDK Examples")
            .build(),
    )
    .send()
    .await;

match response {
    Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
    Err(err) => {
        tracing::info!("Error applying tags to {instance_id}: {err:?}");
        return Err(err.into());
    }
}

tracing::info!("Instance is created.");

Ok(instance_id.to_string())
}

```

- 有关 API 的详细信息，请参阅适用[CreateTags](#)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 SDK 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。

将 **CreateVolume** 与 CLI 配合使用

以下代码示例演示如何使用 CreateVolume。

CLI

AWS CLI

创建空通用型 SSD (gp2) 卷

以下 create-volume 示例将在指定的可用区中创建一个 80 GiB 的通用型 SSD (gp2) 卷。请注意，当前区域必须为 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：从快照创建预调配 IOPS SSD (io1) 卷

以下 create-volume 示例将使用指定的快照在指定的可用区中创建具有 1000 预调配 IOPS 的预调配 IOPS SSD (io1) 卷。

```
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 EBS 加密的默认 CMK 创建加密卷。如果默认情况下禁用了加密，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
  }

```

如果默认情况下启用了加密，则即使没有 `--encrypted` 参数，以下示例命令会创建一个加密卷。

```

aws ec2 create-volume \
  --size 80 \
  --availability-zone us-east-1a

```

如果您使用 `--kms-key-id` 参数指定客户托管的 CMK，则即使默认情况下已启用加密，也必须指定 `--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` 示例将创建一个卷并添加两个标签。

```

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

适用于 PowerShell V4 的工具

示例 1：此示例创建指定的卷。

```
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：此示例请求创建一个卷，并应用键为 stack 且值为 production 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 \(V 4\) CreateVolume](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例创建指定的卷。

```
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：此示例请求创建一个卷，并应用键为 stack 且值为 production 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 \(V 5\) 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 SDK 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。

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：

- [创建具有私有子网和 NAT 网关的 VPC](#)
- [开始使用 Amazon VPC](#)

- [中转网关入门](#)
- [VPC IPAM 入门](#)

CLI

AWS CLI

示例 1：创建 VPC

以下create-vpc示例使用指定 IPv4 CIDR 块和名称标签创建一个 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示例使用指定 IPv4 CIDR 块和专用租户创建一个 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：创建带有 IPv6 CIDR 块的 VPC

以下create-vpc示例使用亚马逊提供的 IPv6 CIDR 块创建一个 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。

PHP

适用于 PHP 的 SDK

Note

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

```
/**  
 * @param string $cidr  
 * @return array  
 */  
public function createVpc(string $cidr): array  
{  
    try {  
        $result = $this->ec2Client->createVpc([  
            "CidrBlock" => $cidr,  
        ]);  
        return $result['Vpc'];  
    } catch (Ec2Exception $caught){  
        echo "There was a problem creating the VPC: {$caught->getAwsErrorMessage()}\n";  
        throw $caught;  
    }  
}
```

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例使用指定的 CIDR 创建 VPC。Amazon VPC 还会为 VPC 创建以下各项：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 (V 4) [CreateVpc](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例使用指定的 CIDR 创建 VPC。Amazon VPC 还会为 VPC 创建以下各项：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 (V 5) [CreateVpc](#) 中的。

Python

适用于 Python 的 SDK (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。

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual
    Private Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def create(self, cidr_block: str) -> str:
        """
        Creates a new Amazon VPC with the specified CIDR block.

        :param cidr_block: The CIDR block for the new VPC, such as '10.0.0.0/16'.
        :return: The ID of the new VPC.
```

```
"""
try:
    response = self.ec2_client.create_vpc(CidrBlock=cidr_block)
    vpc_id = response["Vpc"]["VpcId"]

    waiter = self.ec2_client.get_waiter("vpc_available")
    waiter.wait(VpcIds=[vpc_id])
    return vpc_id
except ClientError as client_error:
    logging.error(
        "Couldn't create the vpc. Here's why: %s",
        client_error.response["Error"]["Message"],
    )
    raise
```

- 有关 API 的详细信息，请参阅适用[CreateVpc](#)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](#) 中查找完整示例，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}'."
  true
rescue StandardError => e
  puts e.message
  false
end

# Example usage:
def run_me
  cidr_block = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-create-vpc.rb ' \
      'CIDR_BLOCK TAG_KEY TAG_VALUE REGION'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-create-vpc.rb ' \
      '10.0.0.0/24 my-key my-value us-west-2'
  end
  exit 1
end

```

```
# 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。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

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

```

" iv_cidr_block = '10.0.0.0/16'
TRY.
    oo_result = lo_ec2->createvpc( iv_cidrblock = iv_cidr_block ).
" oo_result is returned for testing purposes. "
    DATA(lv_vpc_id) = oo_result->get_vpc( )->get_vpcid( ).
    MESSAGE 'Created VPC.' 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[CreateVpc](#)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 SDK 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。

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：

- [创建具有私有子网和 NAT 网关的 VPC](#)

CLI

AWS CLI

示例 1：创建网关端点

以下 create-vpc-endpoint 示例将在 us-east-1 区域中的 VPC vpc-1a2b3c4d 和 Amazon S3 之间创建一个网关 VPC 端点，并将路由表 rtb-11aa22bb 与该端点相关联。

```

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 用户指南》中的[创建网关终端节点](#)。

示例 2：创建接口端点

以下 create-vpc-endpoint 示例将在 us-east-1 区域中的 VPC vpc-1a2b3c4d 和 Amazon S3 之间创建一个接口 VPC 端点。该命令在将子网 subnet-1a2b3c4d 中创建端点，将其与安全组 sg-1a2b3c4d 相关联，并添加一个键为“Service”且值为“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AWS PrivateLink 用户指南中的[创建接口 VPC 终端节点](#)。

示例 3：创建网关负载均衡器端点

以下 create-vpc-endpoint 示例将在 VPC vpc-111122223333aabbcc 和使用网关负载均衡器配置的服务之间创建一个网关负载均衡器端点。

```
aws ec2 create-vpc-endpoint \  
  --service-name com.amazonaws.vpce.us-east-1.vpce-svc-123123a1c43abc123 \  
  --vpc-endpoint-type GatewayLoadBalancer \  
  --vpc-id vpc-111122223333aabbcc \  
  --subnet-ids subnet-0011aabbcc2233445
```

输出：

```
{  
  "VpcEndpoint": {  
    "VpcEndpointId": "vpce-aabbaabbaabbaabba",  
    "VpcEndpointType": "GatewayLoadBalancer",  
    "VpcId": "vpc-111122223333aabbcc",  
    "ServiceName": "com.amazonaws.vpce.us-east-1.vpce-svc-123123a1c43abc123",  
    "State": "pending",  
    "SubnetIds": [  
      "subnet-0011aabbcc2233445"  
    ],  
    "RequesterManaged": false,  
    "NetworkInterfaceIds": [  
      "eni-01010120203030405"  
    ],  
    "CreationTimestamp": "2020-11-11T08:06:03.522Z",  
    "OwnerId": "123456789012"  
  }  
}
```

有关更多信息，请参阅《AWS PrivateLink 用户指南》中的 [Gateway Load Balancer 终端节点](#)。

示例 4：创建资源端点

以下 create-vpc-endpoint 示例创建了一个资源端点。

```
aws ec2 create-vpc-endpoint \  
  --vpc-endpoint-type Resource \  
  --vpc-id vpc-111122223333aabbcc \  
  --subnet-ids subnet-0011aabbcc2233445 \  
  --resource-configuration-arn arn:aws:vpce-lattice-us-east-1:123456789012:resourceconfiguration/rcfg-0123abcde98765432
```

输出：


```
{
  "VpcEndpoint": {
    "VpcEndpointId": "vpce-00939a7ed9EXAMPLE",
    "VpcEndpointType": "Resource",
    "VpcId": "vpc-111122223333aabbcc",
    "State": "Pending",
    "SubnetIds": [
      "subnet-0011aabbcc2233445"
    ],
    "Groups": [
      {
        "GroupId": "sg-03e2f15fbfc09b000",
        "GroupName": "default"
      }
    ],
    "IpAddressType": "IPV4",
    "PrivateDnsEnabled": false,
    "CreationTimestamp": "2025-02-06T23:38:49.525000+00:00",
    "Tags": [],
    "OwnerId": "123456789012",
    "ResourceConfigurationArn": "arn:aws:vpc-lattice:us-east-1:123456789012:resourceconfiguration/rcfg-0123abcde98765432"
  }
}
```

有关更多信息，请参阅《AWS PrivateLink 用户指南》中的[资源端点](#)。

示例 5：创建服务网络端点

以下 create-vpc-endpoint 示例创建了一个服务网络端点。

```
aws ec2 create-vpc-endpoint \
  --vpc-endpoint-type ServiceNetwork \
  --vpc-id vpc-111122223333aabbcc \
  --subnet-ids subnet-0011aabbcc2233445 \
  --service-network-arn arn:aws:vpc-lattice:us-east-1:123456789012:servicenetwork/sn-0101abcd5432abcd0 \
  --security-group-ids sg-0123456789012abcd
```

输出：

```
{
  "VpcEndpoint": {
```

```

    "VpcEndpointId": "vpce-0f00567fa8EXAMPLE",
    "VpcEndpointType": "ServiceNetwork",
    "VpcId": "vpc-111122223333aabbcc",
    "State": "Pending",
    "SubnetIds": [
        "subnet-0011aabbcc2233445"
    ],
    "Groups": [
        {
            "GroupId": "sg-0123456789012abcd",
            "GroupName": "my-security-group"
        }
    ],
    "IpAddressType": "IPv4",
    "PrivateDnsEnabled": false,
    "CreationTimestamp": "2025-02-06T23:44:20.449000+00:00",
    "Tags": [],
    "OwnerId": "123456789012",
    "ServiceNetworkArn": "arn:aws:vpc-lattice:us-east-1:123456789012:servicenetwork/sn-0101abcd5432abcd0"
  }
}

```

有关更多信息，请参阅《AWS PrivateLink 用户指南》中的[服务网络端点](#)。

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

PHP

适用于 PHP 的 SDK

Note

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

```

/**
 * @param string $serviceName
 * @param string $vpcId
 * @param array $routeTableIds

```

```

    * @return array
    */
    public function createVpcEndpoint(string $serviceName, string $vpcId, array
$routeTableIds): array
    {
        try {
            $result = $this->ec2Client->createVpcEndpoint([
                'ServiceName' => $serviceName,
                'VpcId' => $vpcId,
                'RouteTableIds' => $routeTableIds,
            ]);

            return $result["VpcEndpoint"];
        } catch (Ec2Exception $caught){
            echo "There was a problem creating the VPC Endpoint: {$caught-
>getAwsErrorMessage()}\n";
            throw $caught;
        }
    }
}

```

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

PowerShell

适用于 PowerShell V4 的工具

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 端点

```
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 (V 4) [CreateVpcEndpoint](#) 中的。

适用于 PowerShell V5 的工具

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 端点

```
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 \(V 5\) CreateVpcEndpoint](#) 中的。

Python

适用于 Python 的 SDK (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。

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual
    Private Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client
```

```

@classmethod
def from_client(cls) -> "VpcWrapper":
    """
    Creates a VpcWrapper instance with a default EC2 client.

    :return: An instance of VpcWrapper initialized with the default EC2
client.
    """
    ec2_client = boto3.client("ec2")
    return cls(ec2_client)

def create_vpc_endpoint(
    self, vpc_id: str, service_name: str, route_table_ids: list[str]
) -> Dict[str, Any]:
    """
    Creates a new VPC endpoint for the specified service and associates it
with the specified route tables.

    :param vpc_id: The ID of the VPC to create the endpoint in.
    :param service_name: The name of the service to create the endpoint for.
    :param route_table_ids: A list of IDs of the route tables to associate
with the endpoint.
    :return: A dictionary representing the newly created VPC endpoint.
    """
    try:
        response = self.ec2_client.create_vpc_endpoint(
            VpcId=vpc_id,
            ServiceName=service_name,
            RouteTableIds=route_table_ids,
        )
        return response["VpcEndpoint"]
    except ClientError as err:
        logger.error(
            "Couldn't create VPC endpoint for service %s. Here's why: %s:
%s",
            service_name,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise

```

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

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

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

```
" iv_vpc_id = 'vpc-abc123'
" iv_service_name = 'com.amazonaws.region.service'
TRY.
    oo_result = lo_ec2->createvpcendpoint(          " oo_result is
returned for testing purposes. "
    iv_vpcid = iv_vpc_id
    iv_servicename = iv_service_name
    it_routetableids = it_route_table_ids ).
    DATA(lv_vpc_endpoint_id) = oo_result->get_vpcendpoint( )-
>get_vpcendpointid( ).
    MESSAGE 'Created VPC endpoint.' 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[CreateVpcEndpoint](#)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 SDK 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。

将 **CreateVpnConnection** 与 CLI 配合使用

以下代码示例演示如何使用 CreateVpnConnection。

CLI

AWS CLI

示例 1：创建使用动态路由的 VPN 连接

以下 `create-vpn-connection` 示例将在指定的虚拟私有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-001122334455aabbcc \
  --vpn-gateway-id vgw-1a1a1a1a1a1a2b2b2 \
  --tag-specification 'ResourceType=vpn-connection,Tags=[{Key=Name,Value=BGP-VPN}]'
```

输出：

```
{
  "VpnConnection": {
    "CustomerGatewayConfiguration": "...configuration information...",
    "CustomerGatewayId": "cgw-001122334455aabbcc",
    "Category": "VPN",
    "State": "pending",
    "VpnConnectionId": "vpn-123123123123abccab",
    "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《AWS Site-to-Site VPN 用户指南》中的AWS Site-to-Site VPN 工作原理。](#)

示例 2：创建使用静态路由的 VPN 连接

以下 `create-vpn-connection` 示例将在指定的虚拟私有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-001122334455aabbcc \
  --vpn-gateway-id vgw-1a1a1a1a1a1a2b2b2 \
  --options "{\"StaticRoutesOnly\":true}"
```

输出：

```
{
  "VpnConnection": {
    "CustomerGatewayConfiguration": "..configuration information...",
    "CustomerGatewayId": "cgw-001122334455aabbcc",
    "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《AWS Site-to-Site VPN 用户指南》中的AWS Site-to-Site VPN 工作原理。](#)

示例 3：创建 VPN 连接并指定您自己的内部 CIDR 和预共享密钥

以下 create-vpn-connection 示例将创建 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。将在 CustomerGatewayConfiguration 信息中返回指定值。

```
aws ec2 create-vpn-connection \
  --type ipsec.1 \
  --customer-gateway-id cgw-001122334455aabbcc \
  --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-001122334455aabbcc",
    "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《AWS Site-to-Site VPN 用户指南》中的AWS Site-to-Site VPN 工作原理。](#)

示例 4：创建支持 IPv6 流量的 VPN 连接

以下create-vpn-connection示例创建一个 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-001122334455aabbcc \
  --options TunnelInsideIpVersion=ipv6,TunnelOptions=[{StartupAction=start},
{StartupAction=start}]

```

输出：

```

{
  "VpnConnection": {
    "CustomerGatewayConfiguration": "..configuration information...",
    "CustomerGatewayId": "cgw-001122334455aabbcc",
    "Category": "VPN",
    "State": "pending",
    "VpnConnectionId": "vpn-11111111122222222",
    "TransitGatewayId": "tgw-12312312312312312",
    "Options": {
      "EnableAcceleration": false,
      "StaticRoutesOnly": false,
      "LocalIpv6NetworkCidr": "::/0",
      "RemoteIpv6NetworkCidr": "::/0",
      "TunnelInsideIpVersion": "ipv6",
      "TunnelOptions": [

```

```
{
  {
    "OutsideIpAddress": "203.0.113.3",
    "StartupAction": "start"
  },
  {
    "OutsideIpAddress": "203.0.113.5",
    "StartupAction": "start"
  }
],
"Routes": [],
"Tags": []
}
```

有关更多信息，[请参阅《AWS Site-to-Site VPN 用户指南》中的AWS Site-to-Site VPN 工作原理。](#)

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

PowerShell

适用于 PowerShell V4 的工具

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。输出包括网络管理员需要的配置信息（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，并在具有指定名称的文件中捕获配置。

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-
configuration.xml
```

示例 3：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 连接以及静态路由。

```
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 \(V 4\)](#) [CreateVpnConnection](#) 中的。

适用于 PowerShell V5 的工具

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。输出包括网络管理员需要的配置信息（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，并在具有指定名称的文件中捕获配置。

```
(New-EC2VpnConnection -CustomerGatewayId cgw-1a2b3c4d -VpnGatewayId
vgw-1a2b3c4d).CustomerGatewayConfiguration | Out-File C:\path\vpn-
configuration.xml
```

示例 3：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 连接以及静态路由。

```
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 (V 5) [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 SDK 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。

将 **CreateVpnConnectionRoute** 与 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

适用于 PowerShell V4 的工具

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 (V 4) [CreateVpnConnectionRoute](#) 中的。

适用于 PowerShell V5 的工具

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 \(V 5\)](#) [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 SDK 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。

将 **CreateVpnGateway** 与 CLI 配合使用

以下代码示例演示如何使用 CreateVpnGateway。

CLI

AWS CLI

创建虚拟私有网关

此示例创建虚拟私有网关。

命令：

```
aws ec2 create-vpn-gateway --type ipsec.1
```

输出：

```
{
  "VpnGateway": {
    "AmazonSideAsn": 64512,
    "State": "available",
    "Type": "ipsec.1",
    "VpnGatewayId": "vgw-9a4cacf3",
    "VpcAttachments": []
  }
}
```

创建具有特定 Amazon 端 ASN 的虚拟私有网关

此示例创建一个虚拟私有网关，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

适用于 PowerShell V4 的工具

示例 1：此示例创建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 (V 4) [CreateVpnGateway](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例创建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 \(V 5\) 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 SDK 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。

将 **DeleteCustomerGateway** 与 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

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的客户网关。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 \(V 4\) DeleteCustomerGateway](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例删除指定的客户网关。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 \(V 5\) 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 SDK 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。

将 **DeleteDhcpOptions** 与 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

适用于 PowerShell V4 的工具

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 (V 4) [DeleteDhcpOptions](#)中的。

适用于 PowerShell V5 的工具

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 \(V 5\) 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 SDK 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。

将 DeleteFlowLogs 与 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 示例删除指定流日志。

```
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

适用于 PowerShell V4 的工具

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 (V 4) [DeleteFlowLogs](#)中的。

适用于 PowerShell V5 的工具

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 (V 5) [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 SDK 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。

将 `DeleteInternetGateway` 与 CLI 配合使用

以下代码示例演示如何使用 `DeleteInternetGateway`。

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：

- [创建具有私有子网和 NAT 网关的 VPC](#)
- [开始使用 Amazon VPC](#)

CLI

AWS CLI

删除互联网网关

以下 `delete-internet-gateway` 示例删除指定的互联网网关。

```
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

适用于 PowerShell V4 的工具

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 \(V 4\) DeleteInternetGateway](#) 中的。

适用于 PowerShell V5 的工具

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 \(V 5\) 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 SDK 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。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)

- [开始使用 Amazon VPC](#)

.NET

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。

```
/// <summary>
/// Delete an Amazon EC2 key pair.
/// </summary>
/// <param name="keyPairName">The name of the key pair to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteKeyPair(string keyPairName)
{
    try
    {
        await _amazonEC2.DeleteKeyPairAsync(new
DeleteKeyPairRequest(keyPairName)).ConfigureAwait(false);
        return true;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidKeyPair.NotFound")
        {
            _logger.LogError($"KeyPair {keyPairName} does not exist and
cannot be deleted. Please verify the key pair name and try again.");
        }

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

```

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

```

- 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](#) 中查找完整示例，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 getopt command in a function.

```



```

# bashsupport disable=BP5008
function usage() {
    echo "function ec2_delete_keypair"
    echo "Deletes an Amazon EC2 ED25519 or 2048-bit RSA key pair."
    echo "  -n key_pair_name - A key pair name."
    echo ""
}

# Retrieve the calling parameters.
while 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](#) 中查找完整示例，了解如何进行设置和运行。

```
#!/ Delete an Amazon Elastic Compute Cloud (Amazon EC2) instance key pair.
/*!
 \param keyPairName: A name for a key pair.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */

bool AwsDoc::EC2::deleteKeyPair(const Aws::String &keyPairName,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DeleteKeyPairRequest request;

    request.SetKeyName(keyPairName);
    const Aws::EC2::Model::DeleteKeyPairOutcome outcome =
    ec2Client.DeleteKeyPair(
        request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete key pair " << keyPairName <<
            ":" << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted key pair named " << keyPairName <<
            std::endl;
    }

    return outcome.IsSuccess();
}
```

- 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 示例删除指定密钥对。

```
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](#) 中查找完整示例，了解如何进行设置和运行。

```
/**  
 * Deletes a key pair asynchronously.  
 *  
 * @param keyPair the name of the key pair to delete  
 * @return a {@link CompletableFuture} that represents the result of the  
 asynchronous operation.
```

```

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

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

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```

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

/**
 * Deletes the specified key pair, by removing the public key from Amazon EC2.
 * @param {{ keyName: string }} options

```

```
*/
export const main = async ({ keyName }) => {
  const client = new EC2Client({});
  const command = new DeleteKeyPairCommand({
    KeyName: keyName,
  });

  try {
    await client.send(command);
    console.log("Successfully deleted key pair.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "MissingParameter") {
      console.warn(`${caught.message}. Did you provide the required value?`);
    } else {
      throw caught;
    }
  }
};
```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```
suspend fun deleteKeys(keyPair: String?) {
  val request =
    DeleteKeyPairRequest {
      keyName = keyPair
    }

  Ec2Client.fromEnvironment { 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 Kotlin 的 AWS SDK API 参考。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的密钥对。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 (V 4) [DeleteKeyPair](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例删除指定的密钥对。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 \(V 5\) DeleteKeyPair](#) 中的。

Python

适用于 Python 的 SDK (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。

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
        directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is
        stored.
                           This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                           This is a high-level object that wraps key pair actions.
        Optional.
        """
        self.ec2_client = ec2_client
```



```

        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2
client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())

    def delete(self, key_name: str) -> bool:
        """
        Deletes a key pair by its name.

        :param key_name: The name of the key pair to delete.
        :return: A boolean indicating whether the deletion was successful.
        :raises ClientError: If there is an error in deleting the key pair, for
example,
                                if the key pair does not exist.
        """
        try:
            self.ec2_client.delete_key_pair(KeyName=key_name)
            logger.info(f"Successfully deleted key pair: {key_name}")
            self.key_pair = None
            return True
        except self.ec2_client.exceptions.ClientError as err:
            logger.error(f"Deletion failed for key pair: {key_name}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidKeyPair.NotFound":
                logger.error(
                    f"The key pair '{key_name}' does not exist and cannot be
deleted. "
                    "Please verify the key pair name and try again."
                )
            raise

```

- 有关 API 的详细信息，请参阅适用[DeleteKeyPair](#)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](#) 中查找完整示例，了解如何进行设置和运行。

围绕 `delete_key` 的包装器，它也会移除备份的私有 PEM 密钥。

```
pub async fn delete(self, ec2: &EC2, util: &Util) -> Result<(), EC2Error> {
    if let Some(key_name) = self.key_pair.key_name() {
        ec2.delete_key_pair(key_name).await?;
        if let Some(key_path) = self.key_file_path() {
            if let Err(err) = util.remove(key_path) {
                eprintln!("Failed to remove {key_path:?} ({err:?})");
            }
        }
    }
    Ok(())
}
```

```
pub async fn delete_key_pair(&self, key_name: &str) -> Result<(), EC2Error> {
    let key_name: String = key_name.into();
    tracing::info!("Deleting key pair {key_name}");
    self.client
        .delete_key_pair()
        .key_name(key_name)
        .send()
        .await?;
    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用[DeleteKeyPair](#)于 Rust 的AWS SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，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。

Swift

适用于 Swift 的 SDK

Note

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

```
import AWSEC2  
  
/// Delete an EC2 key pair.  
///  
/// - Parameter keyPair: The name of the key pair to delete.
```

```
///  
/// - Returns: `true` if the key pair is deleted successfully; otherwise  
///   `false`.  
func deleteKeyPair(keyPair: String) async -> Bool {  
    do {  
        _ = try await ec2Client.deleteKeyPair(  
            input: DeleteKeyPairInput(  
                keyName: keyPair  
            )  
        )  
  
        return true  
    } catch {  
        print("*** Error deleting the key pair:  
\\(error.localizedDescription)")  
        return false  
    }  
}
```

- 有关 API 的详细信息，请参阅适用于 Swift 的 AWS SDK API 参考 [DeleteKeyPair](#) 中。

有关 AWS SDK 开发者指南和代码示例的完整列表，请参阅 [使用 AWS SDK 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。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](#)
- [创建具有私有子网和 NAT 网关的 VPC](#)

.NET

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。

```
/// <summary>
/// Delete a launch template by name.
/// </summary>
/// <param name="templateName">The name of the template to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteTemplateByName(string templateName)
{
    try
    {
        await _amazonEc2.DeleteLaunchTemplateAsync(
            new DeleteLaunchTemplateRequest()
            {
                LaunchTemplateName = templateName
            });
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode ==
            "InvalidLaunchTemplateName.NotFoundException")
        {
            _logger.LogError(
                $"Could not delete the template, the name
                {_launchTemplateName} was not found.");
        }

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

```
}  
}
```

- 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

此示例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](#) 中查找完整示例，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

适用于 Python 的 SDK (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。

```
class AutoScalingWrapper:  
    """  
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.  
    """  
  
    def __init__(  
        self,  
        resource_prefix: str,  
        inst_type: str,  
        ami_param: str,
```

```

        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

    def delete_template(self):
        """

```



```

Deletes a launch template.
"""
try:
    self.ec2_client.delete_launch_template(
        LaunchTemplateName=self.launch_template_name
    )
    self.delete_instance_profile(
        self.instance_profile_name, self.instance_role_name
    )
    log.info("Launch template %s deleted.", self.launch_template_name)
except ClientError as err:
    if (
        err.response["Error"]["Code"]
        == "InvalidLaunchTemplateName.NotFoundException"
    ):
        log.info(
            "Launch template %s does not exist, nothing to do.",
            self.launch_template_name,
        )
    log.error(f"Full error:\n\t{err}")

```

- 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 SDK 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。

将 **DeleteNetworkAcl** 与 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

适用于 PowerShell V4 的工具

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 (V 4) [DeleteNetworkAcl](#)中的。

适用于 PowerShell V5 的工具

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 \(V 5\) 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 SDK 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。

将 `DeleteNetworkAclEntry` 与 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

适用于 PowerShell V4 的工具

示例 1：此示例从指定的网络 ACL 移除指定的规则。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 \(V 4\) DeleteNetworkAclEntry](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例从指定的网络 ACL 移除指定的规则。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 \(V 5\) 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 SDK 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。

将 **DeleteNetworkInterface** 与 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

适用于 PowerShell V4 的工具

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 (V 4) [DeleteNetworkInterface](#)中的。

适用于 PowerShell V5 的工具

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 \(V 5\) 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 SDK 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。

将 **DeletePlacementGroup** 与 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

此示例命令将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

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的置放群组。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 \(V 4\) DeletePlacementGroup](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例删除指定的置放群组。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 \(V 5\) 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 SDK 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。

将 DeleteRoute 与 CLI 配合使用

以下代码示例演示如何使用 DeleteRoute。

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：

- [中转网关入门](#)

CLI

AWS CLI

删除路由

此示例从指定的路由表中删除指定的路由。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

适用于 PowerShell V4 的工具

示例 1：此示例从指定的路由表中删除指定的路由。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 (V 4) [DeleteRoute](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例从指定的路由表中删除指定的路由。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 (V 5) [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 SDK 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。

将 **DeleteRouteTable** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteRouteTable。

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：

- [创建具有私有子网和 NAT 网关的 VPC](#)
- [开始使用 Amazon VPC](#)

CLI

AWS CLI

删除路由表

此示例删除指定的路由表。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

适用于 PowerShell V4 的工具

示例 1：此示例删除指定的路由表。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 \(V 4\) DeleteRouteTable](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例删除指定的路由表。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 \(V 5\) 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 SDK 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。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)
- [创建具有私有子网和 NAT 网关的 VPC](#)
- [开始使用 Amazon VPC](#)

.NET

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。

```
/// <summary>
/// Delete an Amazon EC2 security group.
/// </summary>
/// <param name="groupName">The name of the group to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteSecurityGroup(string groupId)
{
    try
    {
        var response =
            await _amazonEC2.DeleteSecurityGroupAsync(
                new DeleteSecurityGroupRequest { GroupId = groupId });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidGroup.NotFound")
        {
            _logger.LogError(
                $"Security Group {groupId} does not exist and cannot be
deleted. Please verify the ID and try again.");
        }

        return false;
    }
    catch (Exception ex)
    {

```

```

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

```

- 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](#) 中查找完整示例，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 getopt command in a function.

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

```

```

}

# Retrieve the calling parameters.
while 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](#) 中查找完整示例，了解如何进行设置和运行。

```
///  
//! Delete a security group.  
/*!  
  \param securityGroupID: A security group ID.  
  \param clientConfiguration: AWS client configuration.  
  \return bool: Function succeeded.  
*/  
bool AwsDoc::EC2::deleteSecurityGroup(const Aws::String &securityGroupID,  
                                       const Aws::Client::ClientConfiguration  
                                       &clientConfiguration) {  
    Aws::EC2::EC2Client ec2Client(clientConfiguration);  
    Aws::EC2::Model::DeleteSecurityGroupRequest request;  
  
    request.SetGroupId(securityGroupID);  
    Aws::EC2::Model::DeleteSecurityGroupOutcome outcome =  
    ec2Client.DeleteSecurityGroup(request);  
  
    if (!outcome.IsSuccess()) {  
        std::cerr << "Failed to delete security group " << securityGroupID <<  
        ":" << outcome.GetError().GetMessage() << std::endl;  
    } else {  
        std::cout << "Successfully deleted security group " << securityGroupID <<  
        std::endl;  
    }  
  
    return outcome.IsSuccess();  
}
```

- 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-Classical] 删除安全组

此示例删除名为 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，您不能按名称为 EC2-VPC 引用安全组。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](#) 中查找完整示例，了解如何进行设置和运行。

```
/**
 * Deletes an EC2 security group asynchronously.
 *
 * @param groupId the ID of the security group to delete
 * @return a CompletableFuture that completes when the security group is
 * deleted
```



```

    */
    public CompletableFuture<Void> deleteEC2SecGroupAsync(String groupId) {
        DeleteSecurityGroupRequest request = DeleteSecurityGroupRequest.builder()
            .groupId(groupId)
            .build();

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

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```

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

/**
 * Deletes a security group.
 * @param {{ groupId: string }} options
 */
export const main = async ({ groupId }) => {
    const client = new EC2Client({});
    const command = new DeleteSecurityGroupCommand({
        GroupId: groupId,

```

```
});

try {
    await client.send(command);
    console.log("Security group deleted successfully.");
} catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
        console.warn(`${caught.message}. Please provide a valid GroupId.`);
    } else {
        throw caught;
    }
}
};
```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```
suspend fun deleteEC2SecGroup(groupIdVal: String) {
    val request =
        DeleteSecurityGroupRequest {
            groupId = groupIdVal
        }

    Ec2Client.fromEnvironment { 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 Kotlin 的 AWS SDK API 参考。

PowerShell

适用于 PowerShell V4 的工具

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 \(V 4\) DeleteSecurityGroup](#) 中的。

适用于 PowerShell V5 的工具

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 \(V 5\) DeleteSecurityGroup](#) 中的。

Python

适用于 Python 的 SDK (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。

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

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
    None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
        security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
        high-level identifier
                           that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.
```

```

        :return: An instance of SecurityGroupWrapper initialized with the default
        EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def delete(self, security_group_id: str) -> bool:
        """
        Deletes the specified security group.

        :param security_group_id: The ID of the security group to delete.
        Required.

        :returns: True if the deletion is successful.
        :raises ClientError: If the security group cannot be deleted due to an
        AWS service error.
        """
        try:
            self.ec2_client.delete_security_group(GroupId=security_group_id)
            logger.info(f"Successfully deleted security group
            '{security_group_id}'")
            return True
        except ClientError as err:
            logger.error(f"Deletion failed for security group
            '{security_group_id}'")
            error_code = err.response["Error"]["Code"]

            if error_code == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted
                    because it does not exist."
                )
            elif error_code == "DependencyViolation":
                logger.error(
                    f"Security group '{security_group_id}' cannot be deleted
                    because it is still in use."
                    " Verify that it is:"
                    "\n\t- Detached from resources"
                    "\n\t- Removed from references in other groups"
                    "\n\t- Removed from VPC's as a default group"
                )
            raise

```

- 有关 API 的详细信息，请参阅适用[DeleteSecurityGroup](#)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](#) 中查找完整示例，了解如何进行设置和运行。

```
pub async fn delete_security_group(&self, group_id: &str) -> Result<(),
EC2Error> {
    tracing::info!("Deleting security group {group_id}");
    self.client
        .delete_security_group()
        .group_id(group_id)
        .send()
        .await?;
    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用[DeleteSecurityGroup](#)于 Rust 的AWS SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，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。

Swift

适用于 Swift 的 SDK

Note

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

```

import AWSEC2

/// Delete a security group.
///
/// - Parameter groupId: The ID of the security group to delete.
///
/// - Returns: `true` on successful deletion; `false` on error.
func deleteSecurityGroup(groupId: String) async -> Bool {
    do {
        _ = try await ec2Client.deleteSecurityGroup(
            input: DeleteSecurityGroupInput(
                groupId: groupId
            )
        )

        return true
    } catch {

```

```
        print("*** Error deleting the security group:  
        \(error.localizedDescription)")  
        return false  
    }  
}
```

- 有关 API 的详细信息，请参阅适用于 Swift 的 AWS SDK API 参考 [DeleteSecurityGroup](#) 中。

有关 AWS SDK 开发者指南和代码示例的完整列表，请参阅 [使用 AWS SDK 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

适用于 PowerShell V4 的工具

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 \(V 4\) DeleteSnapshot](#) 中的。

适用于 PowerShell V5 的工具

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 \(V 5\) 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](#) 中查找完整示例，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?;\n\nprintln!("Deleted");\n\nOk(())\n}
```

- 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 SDK 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。

将 **DeleteSpotDatafeedSubscription** 与 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

此示例命令将删除账户的竞价型数据源订阅。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

适用于 PowerShell V4 的工具

示例 1：此示例删除竞价型实例数据源。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 \(V 4\) DeleteSpotDatafeedSubscription](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例删除竞价型实例数据源。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 \(V 5\) 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 SDK 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。

将 DeleteSubnet 与 CLI 配合使用

以下代码示例演示如何使用 DeleteSubnet。

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：

- [创建具有私有子网和 NAT 网关的 VPC](#)

- [开始使用 Amazon VPC](#)

CLI

AWS CLI

删除子网

此示例删除指定的子网。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

适用于 PowerShell V4 的工具

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 (V 4) [DeleteSubnet](#)中的。

适用于 PowerShell V5 的工具

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 (V 5) [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 SDK 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。

将 **DeleteTags** 与 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 示例从指定映像中删除标签 Stack=Test。当您同时指定值和键名称时，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
```

可以选择为标签指定值。以下 delete-tags 示例从指定实例中删除键名称为 purpose 的标签（无论该标签的标签值如何）。

```
aws ec2 delete-tags \
  --resources i-1234567890abcdef0 \
```

```
--tags Key=purpose
```

如果您将空字符串指定为标签值，则仅当标签的值为空字符串时，才会删除标签。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``。如上一个示例所示，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

适用于 PowerShell V4 的工具

示例 1：此示例从指定的资源删除指定的标签，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：此示例从指定的资源删除指定的标签，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：此示例从指定的资源删除指定的标签，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：此示例从指定的资源删除指定的标签，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 (V 4) [DeleteTags](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例从指定的资源删除指定的标签，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：此示例从指定的资源删除指定的标签，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：此示例从指定的资源删除指定的标签，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：此示例从指定的资源删除指定的标签，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 (V 5) [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 SDK 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。

将 **DeleteVolume** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteVolume。

CLI

AWS CLI

删除卷

此示例命令将删除卷 ID 为 vol-049df61146c4d7901 的可用卷。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

适用于 PowerShell V4 的工具

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 (V 4) [DeleteVolume](#) 中的。

适用于 PowerShell V5 的工具

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 (V 5) [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 SDK 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。

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：

- [创建具有私有子网和 NAT 网关的 VPC](#)
- [开始使用 Amazon VPC](#)

- [VPC IPAM 入门](#)

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。

PHP

适用于 PHP 的 SDK

Note

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

```
/**
 * @param string $vpcId
 * @return void
 */
public function deleteVpc(string $vpcId)
{
    try {
        $this->ec2Client->deleteVpc([
            "VpcId" => $vpcId,
        ]);
    } catch (Ec2Exception $caught){
        echo "There was a problem deleting the VPC: {$caught->getAwsErrorMessage()}\n";
    }
}
```

```
        throw $caught;
    }
}
```

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

PowerShell

适用于 PowerShell V4 的工具

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 (V 4) [DeleteVpc](#) 中的。

适用于 PowerShell V5 的工具

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 (V 5) [DeleteVpc](#) 中的。

Python

适用于 Python 的 SDK (Boto3)

Note

还有更多信息 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。

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual
    Private Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
def delete(self, vpc_id: str) -> None:
    """
    Deletes the specified VPC.

    :param vpc_id: The ID of the VPC to delete.
    """
    try:
        self.ec2_client.delete_vpc(VpcId=vpc_id)
    except ClientError as err:
        logger.error(
            "Couldn't delete VPC %s. Here's why: %s: %s",
            vpc_id,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

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

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

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

```
TRY.
    lo_ec2->deletevpc( iv_vpcid = iv_vpc_id ).
    MESSAGE 'Deleted VPC.' 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[DeleteVpc](#)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 SDK 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。

DeleteVpcEndpoints与 AWS SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteVpcEndpoints。

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：

- [创建具有私有子网和 NAT 网关的 VPC](#)

CLI

AWS CLI

删除端点

此示例删除端点 vpce-aa22bb33 和 vpce-1a2b3c4d。如果命令部分成功或不成功，则会返回失败项目的列表。如果命令成功，则返回的列表为空。

命令：

```
aws ec2 delete-vpc-endpoints --vpc-endpoint-ids vpce-aa22bb33 vpce-1a2b3c4d
```

输出：

```
{
  "Unsuccessful": []
}
```

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

PHP

适用于 PHP 的 SDK

Note

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

```
/**
 * @param string $vpcEndpointId
 * @return void
 */
public function deleteVpcEndpoint(string $vpcEndpointId)
{
    try {
        $this->ec2Client->deleteVpcEndpoints([
            "VpcEndpointIds" => [$vpcEndpointId],
        ]);
    } catch (Ec2Exception $caught){
        echo "There was a problem deleting the VPC Endpoint: {$caught-
>getAwsErrorMessage()}\n";
        throw $caught;
    }
}
```

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

Python

适用于 Python 的 SDK (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。

```

class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual
    Private Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def delete_vpc_endpoints(self, vpc_endpoint_ids: list[str]) -> None:
        """
        Deletes the specified VPC endpoints.

        :param vpc_endpoint_ids: A list of IDs of the VPC endpoints to delete.
        """
        try:
            self.ec2_client.delete_vpc_endpoints(VpcEndpointIds=vpc_endpoint_ids)
        except ClientError as err:
            logger.error(
                "Couldn't delete VPC endpoints %s. Here's why: %s: %s",
                vpc_endpoint_ids,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

```


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

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

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

```
TRY.  
    lo_ec2->deletevpcendpoints( it_vpcendpointids = it_vpc_endpoint_ids ).  
    MESSAGE 'Deleted VPC endpoint(s).' 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[DeleteVpcEndpoints](#)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 SDK 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。

将 **DeleteVpnConnection** 与 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

适用于 PowerShell V4 的工具

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 (V 4) [DeleteVpnConnection](#)中的。

适用于 PowerShell V5 的工具

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 \(V 5\)](#) [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 SDK 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。

将 `DeleteVpnConnectionRoute` 与 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

适用于 PowerShell V4 的工具

示例 1：此示例从指定的 VPN 连接中移除指定的静态路由。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 \(V 4\) DeleteVpnConnectionRoute](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例从指定的 VPN 连接中移除指定的静态路由。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 \(V 5\) 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 SDK 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。

将 DeleteVpnGateway 与 CLI 配合使用

以下代码示例演示如何使用 DeleteVpnGateway。

CLI

AWS CLI

删除虚拟私有网关

此示例删除指定的虚拟私有网关。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

适用于 PowerShell V4 的工具

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 (V 4) [DeleteVpnGateway](#)中的。

适用于 PowerShell V5 的工具

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 (V 5) [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 SDK 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。

将 **DeregisterImage** 与 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

适用于 PowerShell V4 的工具

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 (V 4) [DeregisterImage](#) 中的。

适用于 PowerShell V5 的工具

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 \(V 5\) 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 SDK 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。

将 **DescribeAccountAttributes** 与 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 账户的单一属性

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

适用于 PowerShell V4 的工具

示例 1：此示例描述了您是在该区域的 EC2-Classic 和 EC2-VPC 中启动实例，还是只能启动到-VPC 中 EC2。

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

输出 :

```
AttributeValue
-----
EC2
VPC
```

示例 2：此示例描述您的默认 VPC，或者如果您在该区域没有默认 VPC，则为“无”。

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

输出：

```
AttributeValue
-----
vpc-12345678
```

示例 3：此示例描述您可以运行的最大按需型实例数。

```
(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 \(V 4\) DescribeAccountAttributes](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述了您是否可以在该区域的 EC2-Classic 和 EC2-VPC 中启动实例，还是只能启动到-VPC 中 EC2。

```
(Get-EC2AccountAttribute -AttributeName supported-platforms).AttributeValues
```

输出：

```
AttributeValue
-----
EC2
VPC
```

示例 2：此示例描述您的默认 VPC，或者如果您在该区域没有默认 VPC，则为“无”。

```
(Get-EC2AccountAttribute -AttributeName default-vpc).AttributeValues
```

输出：

```
AttributeValue
-----
vpc-12345678
```

示例 3：此示例描述您可以运行的最大按需型实例数。

```
(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 \(V 5\) 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 SDK 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](#) 中查找完整示例，了解如何进行设置和运行。

```
//! Describe all Elastic IP addresses.
/*!
\param clientConfiguration: AWS client configuration.
```

```

    \return bool: Function succeeded.
    */
bool AwsDoc::EC2::describeAddresses(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeAddressesRequest request;
    Aws::EC2::Model::DescribeAddressesOutcome outcome =
    ec2Client.DescribeAddresses(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left << std::setw(20) << "InstanceId" <<
            std::setw(15) << "Public IP" << std::setw(10) << "Domain" <<
            std::setw(30) << "Allocation ID" << std::setw(25) <<
            "NIC ID" << std::endl;

        const Aws::Vector<Aws::EC2::Model::Address> &addresses =
        outcome.GetResult().GetAddresses();
        for (const auto &address: addresses) {
            Aws::String domainString =
                Aws::EC2::Model::DomainTypeMapper::GetNameForDomainType(
                    address.GetDomain());

            std::cout << std::left << std::setw(20) <<
                address.GetInstanceId() << std::setw(15) <<
                address.GetPublicIp() << std::setw(10) << domainString <<
                std::setw(30) << address.GetAllocationId() << std::setw(25)
                << address.GetNetworkInterfaceId() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe Elastic IP addresses:" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- 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：要检索您的 EC2-VPC 弹性 IP 地址的详细信息

以下 `describe-addresses` 示例显示，与 VPC 中的实例结合使用的弹性 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示例显示了与 EC2-VPC 中的特定私有 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：在 EC2-Classic 中检索有关弹性 IP 地址的详细信息

The 以下describe-addresses示例显示了有关您在 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示例显示了带有值的弹性 IP 地址的详细信息203.0.110.25，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](#) 中查找完整示例，了解如何进行设置和运行。

```

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

/**
 * Describes the specified Elastic IP addresses or all of your Elastic IP
 * addresses.
 * @param {{ allocationId: string }} options
 */
export const main = async ({ allocationId }) => {
    const client = new EC2Client({});
    const command = new DescribeAddressesCommand({
        // You can omit this property to show all addresses.
        AllocationIds: [allocationId],
    });

    try {
        const { Addresses } = await client.send(command);
        const addressList = Addresses.map((address) => ` • ${address.PublicIp}`);
        console.log("Elastic IP addresses:");
        console.log(addressList.join("\n"));
    } catch (caught) {
        if (
            caught instanceof Error &&
            caught.name === "InvalidAllocationID.NotFound"
        ) {
            console.warn(`${caught.message}. Please provide a valid AllocationId.`);
        }
    }
}

```



```
    } else {  
        throw caught;  
    }  
}  
};
```

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

PowerShell

适用于 PowerShell V4 的工具

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 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-Classical 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：此示例描述所有弹性 IP 地址。

```
Get-EC2Address
```

示例 6：此示例返回筛选条件中提供的实例 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 Elastic IPs 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
-----
17.212.120.178    eipalloc-012e3b456789e1fad

```

```
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”匹配且值为“Prod”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 \(V 4\) DescribeAddresses](#) 中的。

适用于 PowerShell V5 的工具

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 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：此示例描述所有弹性 IP 地址。

```
Get-EC2Address
```

示例 6：此示例返回筛选条件中提供的实例 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 Elast IPs ic 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
-----
-----
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”匹配且值为“Prod”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 \(V 5\) DescribeAddresses](#) 中的。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

还有更多相关信息 [GitHub](#)。在 [AWS 代码示例存储库](#) 中查找完整示例，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 SDK 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。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](#)
- [创建具有私有子网和 NAT 网关的 VPC](#)
- [开始使用 Amazon VPC](#)
- [中转网关入门](#)

.NET

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。

```
/// <summary>
/// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
/// </summary>
/// <returns>A list of availability zones.</returns>
public async Task<List<string>> DescribeAvailabilityZones()
{
    try
    {
        var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
            new DescribeAvailabilityZonesRequest());
    }
}
```

```

        return zoneResponse.AvailabilityZones.Select(z =>
            z.ZoneName).ToList();
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError($"An error occurred while listing availability
zones.: {ex.Message}");
        throw;
    }
}

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```

//! DescribeAvailabilityZones
/*!
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
int AwsDoc::EC2::describeAvailabilityZones(const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeAvailabilityZonesRequest request;
    Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
    ec2Client.DescribeAvailabilityZones(request);

```



```

    if (outcome.IsSuccess()) {
        std::cout << std::left <<
            std::setw(32) << "ZoneName" <<
            std::setw(20) << "State" <<
            std::setw(32) << "Region" << std::endl;

        const auto &zones =
            outcome.GetResult().GetAvailabilityZones();

        for (const auto &zone: zones) {
            Aws::String stateString =
                Aws::EC2::Model::AvailabilityZoneStateMapper::GetNameForAvailabilityZoneState(
                    zone.GetState());
            std::cout << std::left <<
                std::setw(32) << zone.GetZoneName() <<
                std::setw(20) << stateString <<
                std::setw(32) << zone.GetRegionName() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe availability zones:" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

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

适用于 PowerShell V4 的工具

示例 1：此示例描述当前区域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 \(V 4\) DescribeAvailabilityZones](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述当前区域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 \(V 5\) DescribeAvailabilityZones](#) 中的。

Python

适用于 Python 的 SDK (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。

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

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
```

```

self.ssm_client = ssm_client
self.iam_client = iam_client
sts_client = boto3.client("sts")
self.account_id = sts_client.get_caller_identity()["Account"]

self.key_pair_name = f"{resource_prefix}-key-pair"
self.launch_template_name = f"{resource_prefix}-template-"
self.group_name = f"{resource_prefix}-group"

# Happy path
self.instance_policy_name = f"{resource_prefix}-pol"
self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"

# Failure mode
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

def get_availability_zones(self) -> List[str]:
    """
    Gets a list of Availability Zones in the AWS Region of the Amazon EC2
    client.

    :return: The list of Availability Zones for the client Region.
    """
    try:
        response = self.ec2_client.describe_availability_zones()
        zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
        log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
    except ClientError as err:
        log.error("Failed to retrieve availability zones.")
        log.error(f"Full error:\n\t{err}")
    else:
        return zones

```

- 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

适用于 SAP ABAP 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，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 SAP 的 AWS SDK ABAP API 参考。

有关 SAP AWS SDK 开发者指南和代码示例的完整列表，请参阅[使用 AWS SDK 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。

将 **DescribeBundleTasks** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeBundleTasks。

CLI

AWS CLI

描述您的捆绑任务

此示例描述了您的所有捆绑任务。

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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的捆绑包任务。

```
Get-EC2BundleTask -BundleId bun-12345678
```

示例 2：此示例描述状态为“完成”或“失败”的捆绑包任务。

```
$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 \(V 4\) DescribeBundleTasks](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的捆绑包任务。

```
Get-EC2BundleTask -BundleId bun-12345678
```

示例 2：此示例描述状态为“完成”或“失败”的捆绑包任务。

```
$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 \(V 5\) 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 SDK 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。

将 **DescribeCapacityReservations** 与 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 ",
      "OwnerId": "123456789111",
      "CapacityReservationArn": "arn:aws:ec2:us-east-1:123456789111:capacity-reservation/cr-1234abcd56EXAMPLE",
      "AvailabilityZoneId": "use1-az2",
      "InstanceType": "c5.large",
      "InstancePlatform": "Linux/UNIX",
      "AvailabilityZone": "us-east-1a",
      "Tenancy": "default",
      "TotalInstanceCount": 1,
      "AvailableInstanceCount": 1,
      "EbsOptimized": true,
      "EphemeralStorage": false,
      "State": "active",
      "StartDate": "2024-10-23T15:00:24+00:00",
      "EndDateType": "unlimited",
      "InstanceMatchCriteria": "open",
      "CreateDate": "2024-10-23T15:00:24+00:00",
      "Tags": [],
      "CapacityAllocations": []
    },
    {
      "CapacityReservationId": "cr-abcdEXAMPLE9876ef ",
      "OwnerId": "123456789111",
      "CapacityReservationArn": "arn:aws:ec2:us-east-1:123456789111:capacity-reservation/cr-abcdEXAMPLE9876ef",
      "AvailabilityZoneId": "use1-az2",
      "InstanceType": "c4.large",
      "InstancePlatform": "Linux/UNIX",
      "AvailabilityZone": "us-east-1a",
      "Tenancy": "default",
      "TotalInstanceCount": 1,
      "AvailableInstanceCount": 1,
      "EbsOptimized": true,
      "EphemeralStorage": false,
      "State": "cancelled",
      "StartDate": "2024-10-23T15:01:03+00:00",
      "EndDateType": "unlimited",
      "InstanceMatchCriteria": "open",

```

```

        "CreateDate": "2024-10-23T15:01:02+00:00",
        "Tags": [],
        "CapacityAllocations": []
      }
    ]
  }
}

```

示例 2：描述一个或多个容量预留

以下 `describe-capacity-reservations` 示例显示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-abcdEXAMPLE9876ef ",
      "OwnerId": "123456789111",
      "CapacityReservationArn": "arn:aws:ec2:us-east-1:123456789111:capacity-reservation/cr-abcdEXAMPLE9876ef",
      "AvailabilityZoneId": "use1-az2",
      "InstanceType": "c4.large",
      "InstancePlatform": "Linux/UNIX",
      "AvailabilityZone": "us-east-1a",
      "Tenancy": "default",
      "TotalInstanceCount": 1,
      "AvailableInstanceCount": 1,
      "EbsOptimized": true,
      "EphemeralStorage": false,
      "State": "active",
      "StartDate": "2024-10-23T15:01:03+00:00",
      "EndDateType": "unlimited",
      "InstanceMatchCriteria": "open",
      "CreateDate": "2024-10-23T15:01:02+00:00",
      "Tags": [],
      "CapacityAllocations": []
    }
  ]
}

```

有关更多信息，请参阅《适用于 Linux 实例的 Amazon Elastic Compute Cloud 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

适用于 PowerShell V4 的工具

示例 1：此示例描述该区域的一个或多个容量预留

```
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 (V 4) [DescribeCapacityReservations](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述该区域的一个或多个容量预留

```
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 \(V 5\) 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 SDK 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。

将 **DescribeCustomerGateways** 与 CLI 配合使用

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

CLI

AWS CLI

描述客户网关

此示例描述客户网关。

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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的客户网关。

```
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 \(V 4\) DescribeCustomerGateways](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的客户网关。

```
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 \(V 5\) 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 SDK 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。

将 **DescribeDhcpOptions** 与 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 DHCP 选项

以下 describe-dhcp-options 示例检索有关 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 DHCP 选项并筛选输出

以下 describe-dhcp-options 示例描述了 DHCP 选项，并使用筛选条件仅返回域名服务器具有 example.com 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

适用于 PowerShell V4 的工具

示例 1：此示例列出 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 (V 4) [DescribeDhcpOptions](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例列出 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 \(V 5\) 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 SDK 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。

将 **DescribeFlowLogs** 与 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

适用于 PowerShell V4 的工具

示例 1：此示例描述日志目标类型为“s3”的一个或多个流日志

```
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:::amzn-s3-demo-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 \(V 4\) DescribeFlowLogs](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述日志目标类型为“s3”的一个或多个流日志

```
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:::amzn-s3-demo-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 \(V 5\) 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 SDK 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。

将 `DescribeHostReservationOfferings` 与 CLI 配合使用

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

CLI

AWS CLI

描述专属主机预留产品

此示例描述了可供购买的 M4 实例系列的专属主机预留。

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

适用于 PowerShell V4 的工具

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 \(V 4\) DescribeHostReservationOfferings](#) 中的。

适用于 PowerShell V5 的工具

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 \(V 5\) 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 SDK 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。

将 **DescribeHosts** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeHosts。

CLI

AWS CLI

查看有关专属主机的详细信息

以下describe-hosts示例显示了您 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 Elastic Compute Cloud 用户指南》中的[查看专属主机](#)。

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

PowerShell

适用于 PowerShell V4 的工具

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-01e23f4cd AvailableInstanceCapacity 567899f1 的主机

```
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 \(V 4\) DescribeHosts](#) 中的。

适用于 PowerShell V5 的工具

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-01e23f4cd AvailableInstanceCapacity 567899f1 的主机

```

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 \(V 5\)](#) [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 SDK 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

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。

```
/// <summary>
/// Get the instance profile association data for an instance.
/// </summary>
/// <param name="instanceId">The Id of the instance.</param>
/// <returns>Instance profile associations data.</returns>
public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
{
    try
    {
        var response = await
        _amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
            new DescribeIamInstanceProfileAssociationsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new("instance-id", new List<string>() { instanceId })
                },
            });
        return response.IamInstanceProfileAssociations[0];
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
        {
            _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
        }

        throw;
    }
    catch (Exception ex)
```

```
{
    _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
    throw;
}
```

- 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。

```
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

适用于 Python 的 SDK (Boto3)

Note

还有更多信息 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。


```

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

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"

```

```

self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"

# Failure mode
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
    """
    Gets data about the profile associated with an instance.

    :param instance_id: The ID of the instance to look up.
    :return: The profile data.
    """
    try:
        response =
self.ec2_client.describe_iam_instance_profile_associations(
            Filters=[{"Name": "instance-id", "Values": [instance_id]}]
        )
        if not response["IamInstanceProfileAssociations"]:
            log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
{instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")

```

- 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 SDK 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。

将 **DescribeIdFormat** 与 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，`Deadline` 值表示此资源类型从短 ID 格式永久切换到长 ID 格式的截止时间已于 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。所有曾支持短 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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定资源类型的 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更长时间的资源类型的 ID 格式 IDs。

```
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 \(V 4\) DescribeldFormat](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定资源类型的 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更长时间的资源类型的 ID 格式 IDs。

```
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 \(V 5\)](#) [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 SDK 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。

将 **DescribeIdentityIdFormat** 与 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
```

以下输出表明此角色创建的实例 IDs 以长 ID 格式接收。

```
{
  "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
```

输出表明此用户创建的快照 IDs 以长 ID 格式接收。

```
{
  "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

适用于 PowerShell V4 的工具

示例 1：此示例返回给定角色的资源“image”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 \(V 4\) DescribeIdentityIdFormat](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例返回给定角色的资源“image”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 \(V 5\) 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 SDK 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。

将 **DescribeImageAttribute** 与 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

适用于 PowerShell V4 的工具

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：此示例描述指定 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 \(V 4\) DescribeImageAttribute](#) 中的。

适用于 PowerShell V5 的工具

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：此示例描述指定 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 \(V 5\) 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 SDK 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。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)
- [创建具有私有子网和 NAT 网关的 VPC](#)
- [开始使用 Amazon VPC](#)

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](#) 中查找完整示例，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 getopt command in a function.
```

```

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

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

local aws_cli_args=()

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

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

```

```

    echo "$response"

    return 0
}

```

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-111222333444aaabb",
            "DeleteOnTermination": true,
```

```

        "VolumeType": "gp2",
        "VolumeSize": 10,
        "Encrypted": false
    }
},
"Architecture": "x86_64",
"ImageLocation": "123456789012/RHEL-8.0.0_HVM-20190618-x86_64-1-
Hourly2-GP2",
"RootDeviceType": "ebs",
"OwnerId": "123456789012",
"RootDeviceName": "/dev/sda1",
"CreationDate": "2019-05-10T13:17:12.000Z",
"Public": true,
"ImageType": "machine",
"Name": "RHEL-8.0.0_HVM-20190618-x86_64-1-Hourly2-GP2"
}
]
}

```

有关更多信息，请参阅《[亚马逊 EC2 用户指南](#)》中的[亚马逊系统映像 \(AMI\)](#)。

示例 2：AMIs 根据过滤器进行描述

以下describe-images示例描述了由亚马逊 AMIs 提供的、由亚马逊 EBS 提供支持的 Windows。

```

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。

有关使用筛选条件的其他示例，请参阅 Amazon EC2 用户指南中的[列出和筛选您的资源](#)。

示例 3：AMIs 根据标签进行描述

以下describe-images示例描述了所有带有 AMIs 该标签的内容Type=Custom。该示例使用--query参数仅显示 AMI IDs。

```

aws ec2 describe-images \
  --filters "Name=tag:Type,Values=Custom" \
  --query 'Images[*].[ImageId]' \

```

```
--output text
```

输出：

```
ami-1234567890EXAMPLE  
ami-0abcdef1234567890
```

有关使用标签筛选条件的其他示例，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](#) 中查找完整示例，了解如何进行设置和运行。

```
import { EC2Client, paginateDescribeImages } from "@aws-sdk/client-ec2";  
  
/**  
 * Describes the specified images (AMIs, AKIs, and ARIs) available to you or all  
 * of the images available to you.  
 * @param {{ architecture: string, pageSize: number }} options  
 */  
export const main = async ({ architecture, pageSize }) => {  
  pageSize = Number.parseInt(pageSize);  
  const client = new EC2Client({});  
  
  // The paginate function is a wrapper around the base command.  
  const paginator = paginateDescribeImages(  
    // Without limiting the page size, this call can take a long time. pageSize  
    // is just sugar for  
    // the MaxResults property in the base command.  
    { client, pageSize },  
    {  
      // There are almost 70,000 images available. Be specific with your  
      // filtering  
      // to increase efficiency.
```



```

    // See https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/clients/
    client-ec2/interfaces/describeimagescommandinput.html#filters
    Filters: [{ Name: "architecture", Values: [architecture] }],
  },
);

/**
 * @type {import('@aws-sdk/client-ec2').Image[]}
 */
const images = [];
let recordsScanned = 0;

try {
  for await (const page of paginator) {
    recordsScanned += pageSize;
    if (page.Images.length) {
      images.push(...page.Images);
      break;
    }
    console.log(
      `No matching image found yet. Searched ${recordsScanned} records.`,
    );
  }

  if (images.length) {
    console.log(
      `Found ${images.length} images:\n\n${images.map((image) =>
image.Name).join("\n")}\n`,
    );
  } else {
    console.log(
      `No matching images found. Searched ${recordsScanned} records.\n`,
    );
  }

  return images;
} catch (caught) {
  if (caught instanceof Error && caught.name === "InvalidParameterValue") {
    console.warn(`${caught.message}`);
    return [];
  }
  throw caught;
}
};

```

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的 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 AMIs 了您拥有的。

```
Get-EC2Image -owner self
```

示例 3：此示例描述了运行微软 Windows Server 的公众 AMIs。

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

示例 4：此示例描述了“us-west-2”区域的所有公众 AMIs。

```
Get-EC2Image -Region us-west-2
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeImages](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的 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 AMIs 了您拥有的。

```
Get-EC2Image -owner self
```

示例 3：此示例描述了运行微软 Windows Server 的公众 AMIs。

```
Get-EC2Image -Filter @{ Name="platform"; Values="windows" }
```

示例 4：此示例描述了“us-west-2”区域的所有公众 AMIs。

```
Get-EC2Image -Region us-west-2
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 5\) DescribelImages](#) 中的。

Python

适用于 Python 的 SDK (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。

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
```

```

        :param instances: A list of dictionaries representing Boto3 Instance
        objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
        EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def get_images(self, image_ids: List[str]) -> List[Dict[str, Any]]:
        """
        Gets information about Amazon Machine Images (AMIs) from a list of AMI
        IDs.

        :param image_ids: The list of AMI IDs to look up.
        :return: A list of dictionaries representing the requested AMIs.
        """
        try:
            response = self.ec2_client.describe_images(ImageIds=image_ids)
            images = response["Images"]
        except ClientError as err:
            logger.error(f"Failed to stop AMI(s): {'','.join(map(str,
            image_ids))}")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAMIID.NotFound":
                logger.error("One or more of the AMI IDs does not exist.")
            raise
        return images

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```
pub async fn list_images(&self, ids: Vec<Parameter>) -> Result<Vec<Image>,
EC2Error> {
    let image_ids = ids.into_iter().filter_map(|p| p.value).collect();
    let output = self
        .client
        .describe_images()
        .set_image_ids(Some(image_ids))
        .send()
        .await?;

    let images = output.images.unwrap_or_default();
    if images.is_empty() {
        Err(EC2Error::new("No images for selected AMIs"))
    } else {
        Ok(images)
    }
}
```

将 `list_images` 函数与 SSM 结合使用，以根据环境进行限制。有关 SSM 的更多详细信息，请参阅 https://docs.aws.amazon.com/systems-manager/latest/userguide/example_ss_getparameters_m_section.html。

```
async fn find_image(&mut self) -> Result<ScenarioImage, EC2Error> {
    let params: Vec<Parameter> = self
        .ssm
        .list_path("/aws/service/ami-amazon-linux-latest")
        .await
        .map_err(|e| e.add_message("Could not find parameters for available
images"))?
        .into_iter()
```

```

        .filter(|param| param.name().is_some_and(|name|
name.contains("amzn2"))))
        .collect();
    let amzn2_images: Vec<ScenarioImage> = self
        .ec2
        .list_images(params)
        .await
        .map_err(|e| e.add_message("Could not find images"))?
        .into_iter()
        .map(ScenarioImage::from)
        .collect();
    println!("We will now create an instance from an Amazon Linux 2 AMI");
    let ami = self.util.select_scenario_image(amzn2_images)?;
    Ok(ami)
}

```

- 有关 API 的详细信息，请参阅适用[DescribeImages](#)于 Rust 的 AWS SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

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

```

TRY.
    oo_result = lo_ec2->describeimages( it_imageids = it_image_ids ).
    " oo_result is returned for testing purposes. "
    DATA(lt_images) = oo_result->get_images( ).
    MESSAGE 'Retrieved information about Amazon Machine Images (AMIs).' 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[DescribeImages](#)于 S AP 的AWS SDK ABAP API 参考。

Swift

适用于 Swift 的 SDK

Note

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

```
import AWSEC2

/// Return an array of `EC2ClientTypes.Image` objects describing all of
/// the images in the specified array.
///
/// - Parameter idList: A list of image ID strings indicating the images
///   to return details about.
///
/// - Returns: An array of the images.
func describeImages(_ idList: [String]) async -> [EC2ClientTypes.Image] {
    do {
        let output = try await ec2Client.describeImages(
            input: DescribeImagesInput(
                imageIds: idList
            )
        )

        guard let images = output.images else {
            print("*** No images found.")
            return []
        }

        for image in images {
            guard let id = image.imageId else {
                continue
            }
            print("  \(id): \(image.description ?? "<no description>")")
        }

        return images
    }
}
```



```
    } catch {
      print("*** Error getting image descriptions:
\\(error.localizedDescription)")
      return []
    }
  }
}
```

- 有关 API 的详细信息，请参阅适用于 Swift 的 AWS SDK API 参考 [DescribeImages](#) 中。

有关 AWS SDK 开发者指南和代码示例的完整列表，请参阅 [使用 AWS SDK 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。

将 `DescribeImportImageTasks` 与 CLI 配合使用

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

CLI

AWS CLI

监控导入映像任务

以下 `describe-import-image-tasks` 示例检查指定的导入映像任务的状态。

```
aws ec2 describe-import-image-tasks \
  --import-task-ids import-ami-1234567890abcdef0
```

正在进行的导入映像任务的输出。

```
{
  "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"
  }
]
}

```

已完成的导入映像任务的输出。生成的 AMI 的 ID 由 ImageId 提供。

```

{
  "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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的映像导入任务。

```
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：此示例描述所有映像导入任务。

```
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 \(V 4\) DescribeImportImageTasks](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的映像导入任务。

```
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：此示例描述所有映像导入任务。

```
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 \(V 5\) 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 SDK 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。

将 **DescribeImportSnapshotTasks** 与 CLI 配合使用

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

CLI

AWS CLI

监控导入快照任务

以下 `describe-import-snapshot-tasks` 示例检查指定的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 SnapshotId 提供。

```

{
  "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

适用于 PowerShell V4 的工具

示例 1：此示例描述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 \(V 4\) DescribeImportSnapshotTasks](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述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 \(V 5\) 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 SDK 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。

将 **DescribeInstanceAttribute** 与 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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定实例的实例类型。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

输出：

```
InstanceType           : t2.micro
```

示例 2：此示例描述是否为指定实例启用增强联网。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

输出：

```
SriovNetSupport           : simple
```

示例 3：此示例描述指定实例的安全组。

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

输出：

```
GroupId
-----
sg-12345678
sg-45678901
```

示例 4：此示例描述是否为指定的实例启用 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 \(V 4\) DescribeInstanceAttribute](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定实例的实例类型。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute instanceType
```

输出：

```
InstanceType           : t2.micro
```

示例 2：此示例描述是否为指定实例启用增强联网。

```
Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute sriovNetSupport
```

输出：

```
SriovNetSupport        : simple
```

示例 3：此示例描述指定实例的安全组。

```
(Get-EC2InstanceAttribute -InstanceId i-12345678 -Attribute groupSet).Groups
```

输出：

```
GroupId
-----
sg-12345678
sg-45678901
```

示例 4：此示例描述是否为指定的实例启用 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 (V 5) [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 SDK 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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定实例的状态。

```
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 \(V 4\)](#) [DescribeInstanceStatus](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定实例的状态。

```
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 \(V 5\) 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](#) 中查找完整示例，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 SDK 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：

- [了解基本功能](#)

.NET

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。

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

        var filters = new List<Filter>
        {
```

```

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

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

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

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

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

```

- 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](#) 中查找完整示例，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 Elastic Compute Cloud 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 Elastic Compute Cloud 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](#) 中查找完整示例，了解如何进行设置和运行。

```
/**  
 * Asynchronously retrieves the instance types available in the current AWS  
 region.
```

```

    * <p>
    * This method uses the AWS SDK's asynchronous API to fetch the available
instance types
    * and then processes the response. It logs the memory information, network
information,
    * and instance type for each instance type returned. Additionally, it
returns a
    * {@link CompletableFuture} that resolves to the instance type string for
the "t2.2xlarge"
    * instance type, if it is found in the response. If the "t2.2xlarge"
instance type is not
    * found, an empty string is returned.
    * </p>
    *
    * @return a {@link CompletableFuture} that resolves to the instance type
string for the
    * "t2.2xlarge" instance type, or an empty string if the instance type is not
found
    */
    public CompletableFuture<String> getInstanceTypesAsync() {
        DescribeInstanceTypesRequest typesRequest =
DescribeInstanceTypesRequest.builder()
            .maxResults(10)
            .build();

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

        return response.thenApply(resp -> {

```

```

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

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```

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

/**
 * Describes the specified instance types. By default, all instance types for the
 * current Region are described. Alternatively, you can filter the results.
 * @param {{ pageSize: string, supportedArch: string[], freeTier: boolean }}
 * options
 */
export const main = async ({ pageSize, supportedArch, freeTier }) => {
    pageSize = Number.parseInt(pageSize);
    const client = new EC2Client({});

    // The paginate function is a wrapper around the underlying command.
    const paginator = paginateDescribeInstanceTypes(
        // Without limiting the page size, this call can take a long time. pageSize
        // is just sugar for
        // the MaxResults property in the underlying command.
    );

```

```

    { client, pageSize },
    {
      Filters: [
        {
          Name: "processor-info.supported-architecture",
          Values: supportedArch,
        },
        { Name: "free-tier-eligible", Values: [freeTier ? "true" : "false"] },
      ],
    },
  );

  try {
    /**
     * @type {import('@aws-sdk/client-ec2').InstanceTypeInfo[]}
     */
    const instanceTypes = [];

    for await (const page of paginator) {
      if (page.InstanceTypes.length) {
        instanceTypes.push(...page.InstanceTypes);

        // When we have at least 1 result, we can stop.
        if (instanceTypes.length >= 1) {
          break;
        }
      }
    }

    console.log(
      `Memory size in MiB for matching instance types:\n\n
      ${instanceTypes.map((it) => `${it.InstanceType}: ${it.MemoryInfo.SizeInMiB}
      MiB`).join("\n")}`,
    );
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
      console.warn(`${caught.message}`);
      return [];
    }
    throw caught;
  }
};

```

- 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](#) 中查找完整示例，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.fromEnvironment { 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 Kotlin 的 AWS SDK API 参考。

Python

适用于 Python 的 SDK (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。

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
        objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
        EC2 client.
```

```

    """
    ec2_client = boto3.client("ec2")
    return cls(ec2_client)

    def get_instance_types(
        self, architecture: str = "x86_64", sizes: List[str] = ["*.micro",
        "/*.small"]
    ) -> List[Dict[str, Any]]:
        """
        Gets instance types that support the specified architecture and size.
        See https://docs.aws.amazon.com/AWSEC2/latest/APIReference/
        API_DescribeInstanceTypes.html
        for a list of allowable parameters.

        :param architecture: The architecture supported by instance types.
        Default: 'x86_64'.
        :param sizes: The size of instance types. Default: '/*.micro', '/*.small',
        :return: A list of dictionaries representing instance types that support
        the specified architecture and size.
        """
        try:
            inst_types = []
            paginator = self.ec2_client.get_paginator("describe_instance_types")
            for page in paginator.paginate(
                Filters=[
                    {
                        "Name": "processor-info.supported-architecture",
                        "Values": [architecture],
                    },
                    {"Name": "instance-type", "Values": sizes},
                ]
            ):
                inst_types += page["InstanceTypes"]
        except ClientError as err:
            logger.error(
                f"Failed to get instance types: {architecture},
                {'.'.join(map(str, sizes))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidParameterValue":
                logger.error(
                    "Parameters are invalid. "

```



```

        "Ensure architecture and size strings conform to
DescribeInstanceTypes API reference."
    )
    raise
else:
    return inst_types

```

- 有关 API 的详细信息，请参阅适用[DescribeInstanceTypes](#)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](#) 中查找完整示例，了解如何进行设置和运行。

```

/// List instance types that match an image's architecture and are free tier
eligible.
pub async fn list_instance_types(&self, image: &Image) ->
Result<Vec<InstanceType>, EC2Error> {
    let architecture = format!(
        "{}",
        image.architecture().ok_or_else(|| EC2Error::new(format!(
            "Image {:?} does not have a listed architecture",
            image.image_id()
        )))?
    );
    let free_tier_eligible_filter = Filter::builder()
        .name("free-tier-eligible")
        .values("false")
        .build();
    let supported_architecture_filter = Filter::builder()
        .name("processor-info.supported-architecture")
        .values(architecture)
        .build();

```

```

    let response = self
        .client
        .describe_instance_types()
        .filters(free_tier_eligible_filter)
        .filters(supported_architecture_filter)
        .send()
        .await?;

    Ok(response
        .instance_types
        .unwrap_or_default()
        .into_iter()
        .filter_map(|iti| iti.instance_type)
        .collect())
}

```

- 有关 API 的详细信息，请参阅适用[DescribeInstanceTypes](#)于 Rust 的 AWS SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

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

```

" Create filters for architecture and instance type patterns
" iv_architecture = 'x86_64'
DATA lt_filters TYPE /aws1/cl_ec2filter=>tt_filterlist.
APPEND NEW /aws1/cl_ec2filter(
    iv_name = 'processor-info.supported-architecture'
    it_values = VALUE /aws1/cl_ec2valuestringlist_w=>tt_valuestringlist(
        ( NEW /aws1/cl_ec2valuestringlist_w( iv_architecture ) )
    )
) TO lt_filters.
" Filter for instance type patterns like '*.micro', '*.small'
APPEND NEW /aws1/cl_ec2filter(
    iv_name = 'instance-type'
    it_values = VALUE /aws1/cl_ec2valuestringlist_w=>tt_valuestringlist(

```

```

        ( NEW /aws1/cl_ec2valuestringlist_w( '*.micro' ) )
        ( NEW /aws1/cl_ec2valuestringlist_w( '*.small' ) )
    )
) TO lt_filters.

TRY.
    oo_result = lo_ec2->describeinstancetype( it_filters = lt_filters ).
    " oo_result is returned for testing purposes. "
    DATA(lt_instance_types) = oo_result->get_instancetypes( ).
    MESSAGE 'Retrieved information about EC2 instance types.' 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[DescribeInstanceTypes](#)于 S AP 的 AWS SDK ABAP API 参考。

Swift

适用于 Swift 的 SDK

Note

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

```

import AWSEC2

/// Return a list of instance types matching the specified architecture
/// and instance sizes.
///
/// - Parameters:
///   - architecture: The architecture of the instance types to return, as
///     a member of `EC2ClientTypes.ArchitectureValues`.
///   - sizes: An array of one or more strings identifying sizes of
///     instance type to accept.
///

```

```

    /// - Returns: An array of `EC2ClientTypes.InstanceTypeInfo` records
    ///   describing the instance types matching the given requirements.
    func getMatchingInstanceTypes(architecture: EC2ClientTypes.ArchitectureValues
= EC2ClientTypes.ArchitectureValues.x8664,
                                sizes: [String] = ["*.micro", "*.small"]) async
        -> [EC2ClientTypes.InstanceTypeInfo] {
        var instanceTypes: [EC2ClientTypes.InstanceTypeInfo] = []

        let archFilter = EC2ClientTypes.Filter(
            name: "processor-info.supported-architecture",
            values: [architecture.rawValue]
        )
        let sizeFilter = EC2ClientTypes.Filter(
            name: "instance-type",
            values: sizes
        )

        do {
            let pages = ec2Client.describeInstanceTypesPaginated(
                input: DescribeInstanceTypesInput(
                    filters: [archFilter, sizeFilter]
                )
            )

            for try await page in pages {
                guard let types = page.instanceTypes else {
                    return []
                }

                instanceTypes += types
            }
        } catch {
            print("*** Error getting image types: \(error.localizedDescription)")
            return []
        }

        return instanceTypes
    }

```

- 有关 API 的详细信息，请参阅适用于 Swift 的 AWS SDK API 参考 [DescribeInstanceTypes](#) 中。

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 SDK 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：

- [了解基本功能](#)
- [构建和管理弹性服务](#)
- [开始使用 Amazon VPC](#)

.NET

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。

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

```

        Values = new List<string> { state, },
    },
};
var request = new DescribeInstancesRequest { Filters = filters, };

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

await foreach (var response in paginator.Responses)
{
    foreach (var reservation in response.Reservations)
    {
        foreach (var instance in reservation.Instances)
        {
            Console.Write($"Instance ID: {instance.InstanceId} ");
            Console.WriteLine($"\\tCurrent State:
{instance.State.Name}");
        }
    }
}

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

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

```

- 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](#) 中查找完整示例，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 getopt command in a function.

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

    # Retrieve the calling parameters.
    while getopt "i:q:h" option; do
```

```

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

local aws_cli_args=()

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

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

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

echo "$response"

return 0

```



```
}
```

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](#) 中查找完整示例，了解如何进行设置和运行。

```

//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) instances associated
with an account.
/*!
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeInstances(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeInstancesRequest request;
    bool header = false;
    bool done = false;
    while (!done) {
        Aws::EC2::Model::DescribeInstancesOutcome outcome =
ec2Client.DescribeInstances(request);
        if (outcome.IsSuccess()) {
            if (!header) {
                std::cout << std::left <<
                    std::setw(48) << "Name" <<
                    std::setw(20) << "ID" <<
                    std::setw(25) << "Ami" <<
                    std::setw(15) << "Type" <<
                    std::setw(15) << "State" <<
                    std::setw(15) << "Monitoring" << std::endl;
            }
        }
    }
}

```

```

        header = true;
    }

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

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

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

            Aws::String typeString =

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

            Aws::String monitorString =

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

            const std::vector<Aws::EC2::Model::Tag> &tags =
instance.GetTags();
            auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
                [](const Aws::EC2::Model::Tag
&tag) {
                    return tag.GetKey() ==

"Name";

                });
            if (nameIter != tags.cend()) {
                name = nameIter->GetValue();
            }
            std::cout <<
                std::setw(48) << name <<
                std::setw(20) << instance.GetInstanceId() <<
                std::setw(25) << instance.GetImageId() <<
                std::setw(15) << typeString <<
                std::setw(15) << instanceStateString <<
                std::setw(15) << monitorString << std::endl;

```

```
        }
    }

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

return true;
}
```

- 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 IDs 的实例和子网

以下 describe-instances 示例使用 --query 参数以 JSON 格式仅显示所有实例 IDs 的实例和子网。

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 IDs

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 IDs。

```

aws ec2 describe-instances \
  --filters "Name=instance-type,Values=t2.micro" \
  --query "Reservations[*].Instances[*].[InstanceId]" \
  --output text

```

输出：

```

i-031c0dc19de2fb70c
i-00d8bfff789a736b75
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 IDs、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 and 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-aaaaaaaaa111111111
```

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](#) 中查找完整示例，了解如何进行设置和运行。

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

    DescribeInstancesPublisher paginator =
getAsyncClient().describeInstancesPaginator(request);
    AtomicReference<String> publicIpAddressRef = new AtomicReference<>();
    return paginator.subscribe(response -> {
        response.reservations().stream()
```

```

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

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```

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

/**
 * List all of your EC2 instances running with the provided architecture that
 * were launched in the past month.
 * @param {{ pageSize: string, architectures: string[] }} options
 */
export const main = async ({ pageSize, architectures }) => {
    pageSize = Number.parseInt(pageSize);
    const client = new EC2Client({});
    const d = new Date();

```

```

const year = d.getFullYear();
const month = `0${d.getMonth() + 1}`.slice(-2);
const launchTimePattern = `${year}-${month}-*`;

const paginator = paginateDescribeInstances(
  {
    client,
    pageSize,
  },
  {
    Filters: [
      { Name: "architecture", Values: architectures },
      { Name: "instance-state-name", Values: ["running"] },
      {
        Name: "launch-time",
        Values: [launchTimePattern],
      },
    ],
  },
);

try {
  /**
   * @type {import('@aws-sdk/client-ec2').Instance[]}
   */
  const instanceList = [];
  for await (const page of paginator) {
    const { Reservations } = page;
    for (const reservation of Reservations) {
      instanceList.push(...reservation.Instances);
    }
  }
  console.log(
    `Running instances launched this month:\n\n${instanceList.map((instance) =>
instance.InstanceId).join("\n")}`,
  );
} catch (caught) {
  if (caught instanceof Error && caught.name === "InvalidParameterValue") {
    console.warn(`${caught.message}.`);
  } else {
    throw caught;
  }
}
};

```


- 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](#) 中查找完整示例，了解如何进行设置和运行。

```
suspend fun describeEC2Instances() {
    val request =
        DescribeInstancesRequest {
            maxResults = 6
        }

    Ec2Client.fromEnvironment { 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 Kotlin 的 AWS SDK API 参考。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的实例。

```
(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：此示例描述您在当前区域中的所有实例，按预留分组。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
```

示例 4：此示例说明如何使用具有多个值的筛选器来查询同时处于运行状态和已停止状态的 EC2 实例

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name'; 'Values' = @("running","stopped")}
    )
}

(Get-EC2Instance @InstanceParams).Instances
```

输出：

InstanceId	InstanceType	Platform	PrivateIpAddress	PublicIpAddress
SecurityGroups	SubnetId	VpcId		
i-05a9...f6c5f46e18	t3.medium		10.0.1.7	...
	subnet-1a2b3c4d	vpc-1a2b3c4d		
i-02cf...945c4fdd07	t3.medium	Windows	10.0.1.8	...
	subnet-1a2b3c4d	vpc-1a2b3c4d		
i-0ac0...c037f9f3a1	t3.xlarge	Windows	10.0.1.10	...
	subnet-1a2b3c4d	vpc-1a2b3c4d		
i-066b...57b7b08888	t3.medium	Windows	10.0.1.11	...
	subnet-1a2b3c4d	vpc-1a2b3c4d		
i-0fee...82e83ccd72	t3.medium	Windows	10.0.1.5	...
	subnet-1a2b3c4d	vpc-1a2b3c4d		
i-0a68...274cc5043b	t3.medium	Windows	10.0.1.6	...
	subnet-1a2b3c4d	vpc-1a2b3c4d		

示例 5：此示例说明如何使用具有多个值的过滤器来查询同时运行和已停止的 EC2 实例，以及使用 Select-Object cmdlet 来选择要输出的特定值。

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name'; 'Values' = @("running","stopped")}
    )
}

Select-Object -InputObject ($InstanceParams).Instances -Property InstanceId, InstanceType, Platform, PrivateIpAddress, PublicIpAddress
```

```
$SelectParams = @{
    Property = @(
        "InstanceId", "InstanceType", "Platform", "PrivateIpAddress",
        @{Name="Name";Expression={$_.Tags[$_Tags.Key.IndexOf("Name")].Value}},
        @{Name="State";Expression={$_.State.Name}}
    )
}

$result = Get-EC2Instance @InstanceParams
$result.Instances | Select-Object @SelectParams | Format-Table -AutoSize
```

输出：

InstanceId	InstanceType	Platform	PrivateIpAddress	Name	State
-----	-----	-----	-----	----	-----
i-05a9...f6c5f46e18	t3.medium		10.0.1.7	ec2-name-01	running
i-02cf...945c4fdd07	t3.medium	Windows	10.0.1.8	ec2-name-02	stopped
i-0ac0...c037f9f3a1	t3.xlarge	Windows	10.0.1.10	ec2-name-03	running
i-066b...57b7b08888	t3.medium	Windows	10.0.1.11	ec2-name-04	stopped
i-0fee...82e83ccd72	t3.medium	Windows	10.0.1.5	ec2-name-05	running
i-0a68...274cc5043b	t3.medium	Windows	10.0.1.6	ec2-name-06	stopped

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\)](#) [DescribeInstances](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的实例。

```
(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：此示例描述您在当前区域中的所有实例，按预留分组。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		

示例 4：此示例说明如何使用具有多个值的筛选器来查询同时处于运行状态和已停止状态的 EC2 实例

```

$instanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name';'Values' = @("running","stopped")}
    )
}

(Get-EC2Instance @instanceParams).Instances

```

输出：

InstanceId	InstanceType	Platform	PrivateIpAddress	PublicIpAddress
SecurityGroups	SubnetId	VpcId		

```
-----
-----
i-05a9...f6c5f46e18 t3.medium          10.0.1.7          ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-02cf...945c4fdd07 t3.medium      Windows  10.0.1.8          ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-0ac0...c037f9f3a1 t3.xlarge      Windows  10.0.1.10         ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-066b...57b7b08888 t3.medium      Windows  10.0.1.11         ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-0fee...82e83ccd72 t3.medium      Windows  10.0.1.5          ...
      subnet-1a2b3c4d vpc-1a2b3c4d
i-0a68...274cc5043b t3.medium      Windows  10.0.1.6          ...
      subnet-1a2b3c4d vpc-1a2b3c4d
```

示例 5：此示例说明如何使用具有多个值的过滤器来查询同时运行和已停止的 EC2 实例，以及使用 `Select-Object cmdlet` 来选择要输出的特定值。

```
$InstanceParams = @{
    Filter = @(
        @{'Name' = 'instance-state-name'; 'Values' = @("running", "stopped")}
    )
}

$SelectParams = @{
    Property = @(
        "InstanceID", "InstanceType", "Platform", "PrivateIpAddress",
        @{'Name'="Name"; Expression={$_.Tags[$_].Tags.Key.IndexOf("Name")}.Value}},
        @{'Name'="State"; Expression={$_.State.Name}}
    )
}

$result = Get-EC2Instance @InstanceParams
$result.Instances | Select-Object @SelectParams | Format-Table -AutoSize
```

输出：

InstanceId	InstanceType	Platform	PrivateIpAddress	Name	State
-----	-----	-----	-----	----	-----
i-05a9...f6c5f46e18	t3.medium		10.0.1.7	ec2-name-01	running
i-02cf...945c4fdd07	t3.medium	Windows	10.0.1.8	ec2-name-02	stopped
i-0ac0...c037f9f3a1	t3.xlarge	Windows	10.0.1.10	ec2-name-03	running
i-066b...57b7b08888	t3.medium	Windows	10.0.1.11	ec2-name-04	stopped

i-0fee...82e83ccd72	t3.medium	Windows	10.0.1.5	ec2-name-05	running
i-0a68...274cc5043b	t3.medium	Windows	10.0.1.6	ec2-name-06	stopped

示例 6：此示例验证了在不实际获取 EC2 实例的情况下使用 DryRun 参数获取实例的权限。
注：这在操作成功时会引发异常，这是预期行为。

```
Get-EC2Tag -DryRun $true
```

输出：

```
Get-EC2Instance: Request would have succeeded, but DryRun flag is set.
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 5\) DescribeInstances](#) 中的。

Python

适用于 Python 的 SDK (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。

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
```

```

        :param instances: A list of dictionaries representing Boto3 Instance
        objects. These are high-level objects that
                           wrap instance actions.

        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
        EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def display(self, state_filter: Optional[str] = "running") -> None:
        """
        Displays information about instances, filtering by the specified state.

        :param state_filter: The instance state to include in the output. Only
        instances in this state
                           will be displayed. Default is 'running'. Example
        states: 'running', 'stopped'.
        """
        if not self.instances:
            logger.info("No instances to display.")
            return

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        paginator = self.ec2_client.get_paginator("describe_instances")
        page_iterator = paginator.paginate(InstanceIds=instance_ids)

        try:
            for page in page_iterator:
                for reservation in page["Reservations"]:
                    for instance in reservation["Instances"]:
                        instance_state = instance["State"]["Name"]

                        # Apply the state filter (default is 'running')
                        if state_filter and instance_state != state_filter:

```

```

        continue # Skip this instance if it doesn't match
the filter

        # Create a formatted string with instance details
        instance_info = (
            f"• ID: {instance['InstanceId']}\n"
            f"• Image ID: {instance['ImageId']}\n"
            f"• Instance type: {instance['InstanceType']}\n"
            f"• Key name: {instance['KeyName']}\n"
            f"• VPC ID: {instance['VpcId']}\n"
            f"• Public IP: {instance.get('PublicIpAddress', 'N/A')}\n"

            f"• State: {instance_state}"
        )
        print(instance_info)

    except ClientError as err:
        logger.error(
            f"Failed to display instance(s). : {' '.join(map(str,
instance_ids)))}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidInstanceID.NotFound":
            logger.error(
                "One or more instance IDs do not exist. "
                "Please verify the instance IDs and try again."
            )
            raise

```

- 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](#) 中查找完整示例，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
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](#) 中查找完整示例，了解如何进行设置和运行。

检索 EC2 实例的详细信息。

```

pub async fn describe_instance(&self, instance_id: &str) -> Result<Instance,
EC2Error> {
    let response = self
        .client
        .describe_instances()
        .instance_ids(instance_id)
        .send()
        .await?;

    let instance = response
        .reservations()
        .first()
        .ok_or_else(|| EC2Error::new(format!("No instance reservations for
{instance_id}")))?
        .instances()
        .first()

```

```

        .ok_or_else(|| {
            EC2Error::new(format!("No instances in reservation for
{instance_id}"))
        })?;

        Ok(instance.clone())
    }

```

创建 EC2 实例后，检索并存储其详细信息。

```

/// Create an EC2 instance with the given ID on a given type, using a
/// generated KeyPair and applying a list of security groups.
pub async fn create(
    &mut self,
    ec2: &EC2,
    image_id: &str,
    instance_type: InstanceType,
    key_pair: &KeyPairInfo,
    security_groups: Vec<&SecurityGroup>,
) -> Result<(), EC2Error> {
    let instance_id = ec2
        .create_instance(image_id, instance_type, key_pair, security_groups)
        .await?;
    let instance = ec2.describe_instance(&instance_id).await?;
    self.instance = Some(instance);
    Ok(())
}

```

- 有关 API 的详细信息，请参阅适用[DescribeInstances](#)于 Rust 的 AWS SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，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_instance_id    TYPE /aws1/ec2string,
           lv_status         TYPE /aws1/ec2instancename,
           lv_instance_type  TYPE /aws1/ec2instancetype,
           lv_image_id       TYPE /aws1/ec2string.
    LOOP AT oo_result->get_reservations( ) INTO DATA(lo_reservation).
        LOOP AT lo_reservation->get_instances( ) INTO DATA(lo_instance).
            lv_instance_id = lo_instance->get_instanceid( ).
            lv_status = lo_instance->get_state( )->get_name( ).
            lv_instance_type = lo_instance->get_instancetype( ).
            lv_image_id = lo_instance->get_imageid( ).
        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 SDK 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。

将 **DescribeInternetGateways** 与 CLI 配合使用

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

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 Amazon VPC](#)

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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的互联网网关。


```
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 \(V 4\) DescribeInternetGateways](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的互联网网关。

```
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 \(V 5\) 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 SDK 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：

- [了解基本功能](#)

.NET

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。

```

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

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

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

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

- 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](#) 中查找完整示例，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 getopt command in a function.

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

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

    local response
```

```

response=$(aws ec2 describe-key-pairs \
  --query 'KeyPairs[*].[KeyName, KeyFingerprint]' \
  --output text) || {
  aws_cli_error_log ${?}
  errecho "ERROR: AWS reports describe-key-pairs operation failed.$response"
  return 1
}

echo "$response"

return 0
}

```

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](#) 中查找完整示例，了解如何进行设置和运行。

```

//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) instance key pairs.
/*!
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::describeKeyPairs(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeKeyPairsRequest request;

    Aws::EC2::Model::DescribeKeyPairsOutcome outcome =
    ec2Client.DescribeKeyPairs(request);
    if (outcome.IsSuccess()) {

```

```

        std::cout << std::left <<
            std::setw(32) << "Name" <<
            std::setw(64) << "Fingerprint" << std::endl;

        const std::vector<Aws::EC2::Model::KeyPairInfo> &key_pairs =
            outcome.GetResult().GetKeyPairs();
        for (const auto &key_pair: key_pairs) {
            std::cout << std::left <<
                std::setw(32) << key_pair.GetKeyName() <<
                std::setw(64) << key_pair.GetKeyFingerprint() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe key pairs:" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```

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

    return responseFuture;
}

```


- 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](#) 中查找完整示例，了解如何进行设置和运行。

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

/**
 * List all key pairs in the current AWS account.
 * @param {{ dryRun: boolean }}
 */
export const main = async ({ dryRun }) => {
  const client = new EC2Client({});
  const command = new DescribeKeyPairsCommand({ DryRun: dryRun });

  try {
    const { KeyPairs } = await client.send(command);
    const keyPairList = KeyPairs.map(
      (kp) => ` • ${kp.KeyPairId}: ${kp.KeyName}`,
    ).join("\n");
    console.log("The following key pairs were found in your account:");
    console.log(keyPairList);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "DryRunOperation") {
      console.log(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```
suspend fun describeEC2Keys() {  
    Ec2Client.fromEnvironment { 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 Kotlin 的 AWS SDK API 参考。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的密钥对。

```
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 \(V 4\) DescribeKeyPairs](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的密钥对。

```
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 \(V 5\) DescribeKeyPairs](#) 中的。

Python

适用于 Python 的 SDK (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。

```
class KeyPairWrapper:
    """
    Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) key pair actions.
    This class provides methods to create, list, and delete EC2 key pairs.
    """

    def __init__(
        self,
        ec2_client: boto3.client,
```

```

        key_file_dir: Union[tempfile.TemporaryDirectory, str],
        key_pair: Optional[dict] = None,
    ):
        """
        Initializes the KeyPairWrapper with the specified EC2 client, key file
        directory,
        and an optional key pair.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        :param key_file_dir: The folder where the private key information is
        stored.
                           This should be a secure folder.
        :param key_pair: A dictionary representing the Boto3 KeyPair object.
                           This is a high-level object that wraps key pair actions.
        Optional.
        """
        self.ec2_client = ec2_client
        self.key_pair = key_pair
        self.key_file_path: Optional[str] = None
        self.key_file_dir = key_file_dir

    @classmethod
    def from_client(cls) -> "KeyPairWrapper":
        """
        Class method to create an instance of KeyPairWrapper using a new EC2
client
        and a temporary directory for storing key files.

        :return: An instance of KeyPairWrapper.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client, tempfile.TemporaryDirectory())

    def list(self, limit: Optional[int] = None) -> None:
        """
        Displays a list of key pairs for the current account.

        WARNING: Results are not paginated.

        :param limit: The maximum number of key pairs to list. If not specified,
all key pairs will be listed.

```

```

        :raises ClientError: If there is an error in listing the key pairs.
        """
        try:
            response = self.ec2_client.describe_key_pairs()
            key_pairs = response.get("KeyPairs", [])

            if limit:
                key_pairs = key_pairs[:limit]

            for key_pair in key_pairs:
                logger.info(
                    f"Found {key_pair['KeyType']} key '{key_pair['KeyName']}'
with fingerprint:"
                )
                logger.info(f"\t{key_pair['KeyFingerprint']}")
        except ClientError as err:
            logger.error(f"Failed to list key pairs: {str(err)}")
            raise

```

- 有关 API 的详细信息，请参阅适用[DescribeKeyPairs](#)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](#) 中查找完整示例，了解如何进行设置和运行。

```

pub async fn list_key_pair(&self) -> Result<Vec<KeyPairInfo>, EC2Error> {
    let output = self.client.describe_key_pairs().send().await?;
    Ok(output.key_pairs.unwrap_or_default())
}

```

- 有关 API 的详细信息，请参阅适用[DescribeKeyPairs](#)于 Rust 的AWS SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，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。

Swift

适用于 Swift 的 SDK

Note

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

```
import AWSEC2

/// Describe the key pairs associated with the user by outputting each key
/// pair's name and fingerprint.
func describeKeyPairs() async {
    do {
```

```

        let output = try await ec2Client.describeKeyPairs(
            input: DescribeKeyPairsInput()
        )

        guard let keyPairs = output.keyPairs else {
            print("*** No key pairs list available.")
            return
        }

        for keyPair in keyPairs {
            print(keyPair.keyName ?? "<unknown>", ":",
                keyPair.keyFingerprint ?? "<unknown>")
        }
    } catch {
        print("*** Error: Unable to obtain a key pair list.")
    }
}

```

- 有关 API 的详细信息，请参阅适用于 Swift 的 AWS SDK API 参考 [DescribeKeyPairs](#) 中。

有关 AWS SDK 开发者指南和代码示例的完整列表，请参阅 [使用 AWS SDK 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。

将 **DescribeNetworkAcls** 与 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 ACLs

以下 `describe-network-acls` 示例检索有关您的网络 ACLs 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》ACLs 中的[网络](#)。

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的网络 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 ACLs。

```
Get-EC2NetworkAcl
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeNetworkAcls](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的网络 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 ACLs。

```
Get-EC2NetworkAcl
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 5\) 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 SDK 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。

将 `DescribeNetworkInterfaceAttribute` 与 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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的网络接口。

```
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 \(V 4\) DescribeNetworkInterfaceAttribute](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的网络接口。

```
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 \(V 5\) 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 SDK 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。

将 **DescribeNetworkInterfaces** 与 CLI 配合使用

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

CLI

AWS CLI

描述网络接口

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"
    }
]

```

```
}
```

此示例描述了具有带有键 Purpose 和值 Prod 的标签的网络接口。

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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的网络接口。

```
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 \(V 4\) DescribeNetworkInterfaces](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的网络接口。

```
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 \(V 5\) 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 SDK 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。

将 **DescribePlacementGroups** 与 CLI 配合使用

以下代码示例演示如何使用 DescribePlacementGroups。

CLI

AWS CLI

描述置放群组

此示例命令描述了所有置放群组。

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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的置放群组。

```
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 \(V 4\) DescribePlacementGroups](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的置放群组。

```
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 \(V 5\) 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 SDK 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。

将 **DescribePrefixLists** 与 CLI 配合使用

以下代码示例演示如何使用 `DescribePrefixLists`。

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：

- [创建具有私有子网和 NAT 网关的 VPC](#)

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

适用于 PowerShell V4 的工具

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 (V 4) [DescribePrefixLists](#)中的。

适用于 PowerShell V5 的工具

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 \(V 5\)](#) [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 SDK 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](#) 中查找完整示例，了解如何进行设置和运行。

```
#!/usr/bin/perl
use strict;
use warnings;

use Amazon::ElasticComputeCloud;

my $client = Amazon::ElasticComputeCloud->new(
    \param clientConfiguration: AWS client configuration.
);

my $regions = $client->describe_regions();

\return bool: Function succeeded.
```

```

*/
bool AwsDoc::EC2::describeRegions(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::DescribeRegionsRequest request;
    Aws::EC2::Model::DescribeRegionsOutcome outcome =
    ec2Client.DescribeRegions(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
            std::setw(32) << "RegionName" <<
            std::setw(64) << "Endpoint" << std::endl;

        const auto &regions = outcome.GetResult().GetRegions();
        for (const auto &region: regions) {
            std::cout << std::left <<
                std::setw(32) << region.GetRegionName() <<
                std::setw(64) << region.GetEndpoint() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe regions:" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    std::cout << std::endl;

    return outcome.IsSuccess();
}

```

- 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 [区域和区域](#)。

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 [区域和区域](#)。

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 [区域和区域](#)。

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[区域和区域](#)。

- 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](#) 中查找完整示例，了解如何进行设置和运行。

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

/**
 * List all available AWS regions.
 * @param {{ regionNames: string[], includeOptInRegions: boolean }} options
 */
export const main = async ({ regionNames, includeOptInRegions }) => {
  const client = new EC2Client({});
  const command = new DescribeRegionsCommand({
    // By default this command will not show regions that require you to opt-in.
    // When AllRegions is true, even the regions that require opt-in will be
    returned.
    AllRegions: includeOptInRegions,
    // You can omit the Filters property if you want to get all regions.
    Filters: regionNames?.length
      ? [
          {
            Name: "region-name",
```

```

        // You can specify multiple values for a filter.
        // You can also use '*' as a wildcard. This will return all
        // of the regions that start with `us-east-`.
        Values: regionNames,
    },
]
: undefined,
});

try {
    const { Regions } = await client.send(command);
    const regionsList = Regions.map((reg) => ` • ${reg.RegionName}`);
    console.log("Found regions:");
    console.log(regionsList.join("\n"));
} catch (caught) {
    if (caught instanceof Error && caught.name === "DryRunOperation") {
        console.log(`${caught.message}`);
    } else {
        throw caught;
    }
}
};

```

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例描述可供您使用的区域。

```
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 \(V 4\) DescribeRegions](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述可供您使用的区域。

```
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 \(V 5\) 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](#) 中查找完整示例，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](#) 中查找完整示例，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

适用于 SAP ABAP 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，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 SDK 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。

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：

- [开始使用 Amazon VPC](#)
- [中转网关入门](#)

CLI

AWS CLI

描述路由表

以下 `describe-route-tables` 示例检索有关路由表的详细信息。

```
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。

PHP

适用于 PHP 的 SDK

Note

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

```

/**
 * @param array $routeTableIds
 * @param array $filters
 * @return array
 */
public function describeRouteTables(array $routeTableIds = [], array $filters
= []): array
{
    $parameters = [];
    if($routeTableIds){
        $parameters['RouteTableIds'] = $routeTableIds;
    }
}

```

```

        if($filters){
            $parameters['Filters'] = $filters;
        }
        try {
            $paginator = $this->ec2Client->getPaginator("DescribeRouteTables",
            $parameters);
            $contents = [];
            foreach ($paginator as $result) {
                foreach ($result['RouteTables'] as $object) {
                    $contents[] = $object['RouteTableId'];
                }
            }
        }catch (Ec2Exception $caught){
            echo "There was a problem paginating the results of
            DescribeRouteTables: {$caught->getAwsErrorMessage()}\n";
            throw $caught;
        }
        return $contents;
    }
}

```

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例描述所有路由表。

```
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 \(V 4\) DescribeRouteTables](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述所有路由表。

```
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 \(V 5\) DescribeRouteTables](#) 中的。

Python

适用于 Python 的 SDK (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。

```
class VpcWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Amazon Virtual
    Private Cloud actions."""

    def __init__(self, ec2_client: boto3.client):
        """
        Initializes the VpcWrapper with an EC2 client.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                           access to AWS EC2 services.
        """
        self.ec2_client = ec2_client

    @classmethod
    def from_client(cls) -> "VpcWrapper":
        """
        Creates a VpcWrapper instance with a default EC2 client.

        :return: An instance of VpcWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def describe_route_tables(self, vpc_ids: list[str]) -> None:
        """
        Displays information about the route tables in the specified VPC.

        :param vpc_ids: A list of VPC IDs.
        """
```

```

try:
    response = self.ec2_client.describe_route_tables(
        Filters=[{"Name": "vpc-id", "Values": vpc_ids}]
    )
    pp(response["RouteTables"])
except ClientError as err:
    logger.error(
        "Couldn't describe route tables for VPCs %s. Here's why: %s: %s",
        vpc_ids,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise

```

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

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

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

```

" Create filter for VPC ID
" iv_vpc_id = 'vpc-abc123'
DATA lt_filters TYPE /aws1/cl_ec2filter=>tt_filterlist.
APPEND NEW /aws1/cl_ec2filter(
    iv_name = 'vpc-id'
    it_values = VALUE /aws1/cl_ec2valuestringlist_w=>tt_valuestringlist(
        ( NEW /aws1/cl_ec2valuestringlist_w( iv_vpc_id ) )
    )
) TO lt_filters.

TRY.

```



```

        oo_result = lo_ec2->describeroutetables( it_filters = lt_filters ).
        " oo_result is returned for testing purposes. "
        DATA(lt_route_tables) = oo_result->get_routetables( ).
        MESSAGE 'Retrieved information about route tables.' 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[DescribeRouteTables](#)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 SDK 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。

将 **DescribeScheduledInstanceAvailability** 与 CLI 配合使用

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

CLI

AWS CLI

描述有效的计划

此示例描述了从指定日期开始每周星期日发生的计划。

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，您可以添加筛选条件，用于指定操作系统、网络 and 实例类型。

命令:

```
--filters name=Platform , values=linux/UNIX 名称=网络平台 , 值=-VPC 名称=实例类型 , values=c4.large EC2
```

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例描述从指定日期开始每周星期日发生的计划。

```

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 and 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 \(V 4\) DescribeScheduledInstanceAvailability](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述从指定日期开始每周星期日发生的计划。

```
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 and 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 \(V 5\) 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 SDK 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。

将 **DescribeScheduledInstances** 与 CLI 配合使用

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

CLI

AWS CLI

描述计划实例

此示例描述了指定的计划实例。

命令：

```
aws ec2 describe-scheduled-instances --scheduled-instance-
ids sci-1234-1234-1234-123456789012
```

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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的计划实例。

```
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 \(V 4\) DescribeScheduledInstances](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的计划实例。

```
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 \(V 5\) 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 SDK 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。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)
- [开始使用 Amazon VPC](#)

.NET

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。

```

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

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

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

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

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

            throw;
        }
        catch (Exception ex)
        {

```



```

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

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

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

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

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

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

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

```

```

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

        Console.WriteLine($"\\n\\tPrefixListIds: ");
        permission.PrefixListIds.ForEach(id => Console.Write($"{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](#) 中查找完整示例，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 getopt command in a function.

```

```

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

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

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

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

local error_code=${?}

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

```

```

    return 1
fi

echo "$response"

return 0
}

```

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."
    fi
}

```

```

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](#) 中查找完整示例，了解如何进行设置和运行。

```

//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) security groups, or a
    specific group.
/*!
    \param groupID: A group ID, ignored if empty.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::describeSecurityGroups(const Aws::String &groupID,
                                         const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeSecurityGroupsRequest request;

    if (!groupID.empty()) {
        request.AddGroupIds(groupID);
    }

    Aws::String nextToken;

```

```

do {
    if (!nextToken.empty()) {
        request.SetNextToken(nextToken);
    }

    Aws::EC2::Model::DescribeSecurityGroupsOutcome outcome =
ec2Client.DescribeSecurityGroups(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
            std::setw(32) << "Name" <<
            std::setw(30) << "GroupId" <<
            std::setw(30) << "VpcId" <<
            std::setw(64) << "Description" << std::endl;

        const std::vector<Aws::EC2::Model::SecurityGroup> &securityGroups =
            outcome.GetResult().GetSecurityGroups();

        for (const auto &securityGroup: securityGroups) {
            std::cout << std::left <<
                std::setw(32) << securityGroup.GetGroupName() <<
                std::setw(30) << securityGroup.GetGroupId() <<
                std::setw(30) << securityGroup.GetVpcId() <<
                std::setw(64) << securityGroup.GetDescription() <<
                std::endl;
        }
    } else {
        std::cerr << "Failed to describe security groups:" <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    nextToken = outcome.GetResult().GetNextToken();
} while (!nextToken.empty());

return true;
}

```

- 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` 示例使用筛选条件将结果范围限定为具有允许 SSH 流量（端口 22）的规则和允许来自所有地址的流量（0.0.0.0/0）的规则的安全组。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。该示例使用 `--query` 参数仅显示安全组 IDs 的名称和名称。

```
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"
  }
]
```

有关使用标签筛选条件的其他示例，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](#) 中查找完整示例，了解如何进行设置和运行。

```

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

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

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

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

/**
 * Describes the specified security groups or all of your security groups.
 * @param {{ groupIds: string[] }} options
 */
export const main = async ({ groupIds = [] }) => {
  const client = new EC2Client({});
  const command = new DescribeSecurityGroupsCommand({
    GroupIds: groupIds,
  });

  try {
    const { SecurityGroups } = await client.send(command);
    const sgList = SecurityGroups.map(
      (sg) => `• ${sg.GroupName} (${sg.GroupId}): ${sg.Description}`,
    ).join("\n");
    if (sgList.length) {
      console.log(`Security groups:\n${sgList}`);
    } else {
      console.log("No security groups found.");
    }
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidGroupId.Malformed") {
      console.warn(`${caught.message}. Please provide a valid GroupId.`);
    } else if (
      caught instanceof Error &&
      caught.name === "InvalidGroup.NotFound"
    ) {

```

```
    ) {  
        console.warn(caught.message);  
    } else {  
        throw caught;  
    }  
}  
};
```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```
suspend fun describeEC2SecurityGroups(groupId: String) {  
    val request =  
        DescribeSecurityGroupsRequest {  
            groupIds = listOf(groupId)  
        }  
  
    Ec2Client.fromEnvironment { 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 Kotlin 的 AWS SDK API 参考。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例描述 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。在 EC2-Classic 中使用安全组时，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 \(V 4\) DescribeSecurityGroups](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述 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。在 EC2-Classic 中使用安全组时，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 \(V 5\) DescribeSecurityGroups](#) 中的。

Python

适用于 Python 的 SDK (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。

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

    def __init__(self, ec2_client: boto3.client, security_group: Optional[str] =
    None):
        """
        Initializes the SecurityGroupWrapper with an EC2 client and an optional
        security group ID.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param security_group: The ID of a security group to manage. This is a
        high-level identifier
                           that represents the security group.
        """
        self.ec2_client = ec2_client
        self.security_group = security_group

    @classmethod
    def from_client(cls) -> "SecurityGroupWrapper":
        """
        Creates a SecurityGroupWrapper instance with a default EC2 client.

        :return: An instance of SecurityGroupWrapper initialized with the default
        EC2 client.
        """
```

```

    ec2_client = boto3.client("ec2")
    return cls(ec2_client)

    def describe(self, security_group_id: Optional[str] = None) -> bool:
        """
        Displays information about the specified security group or all security
        groups if no ID is provided.

        :param security_group_id: The ID of the security group to describe.
                                   If None, an open search is performed to
        describe all security groups.
        :returns: True if the description is successful.
        :raises ClientError: If there is an error describing the security
        group(s), such as an invalid security group ID.
        """
        try:
            paginator = self.ec2_client.get_paginator("describe_security_groups")

            if security_group_id is None:
                # If no ID is provided, return all security groups.
                page_iterator = paginator.paginate()
            else:
                page_iterator = paginator.paginate(GroupIds=[security_group_id])

            for page in page_iterator:
                for security_group in page["SecurityGroups"]:
                    print(f"Security group: {security_group['GroupName']}")
                    print(f"\tID: {security_group['GroupId']}")
                    print(f"\tVPC: {security_group['VpcId']}")
                    if security_group["IpPermissions"]:
                        print("Inbound permissions:")
                        pp(security_group["IpPermissions"])

            return True
        except ClientError as err:
            logger.error("Failed to describe security group(s).")
            if err.response["Error"]["Code"] == "InvalidGroup.NotFound":
                logger.error(
                    f"Security group {security_group_id} does not exist "
                    f"because the specified security group ID was not found."
                )
            raise

```


- 有关 API 的详细信息，请参阅适用[DescribeSecurityGroups](#)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](#) 中查找完整示例，了解如何进行设置和运行。

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

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

```

    }
  }
}

```

- 有关 API 的详细信息，请参阅适用[DescribeSecurityGroups](#)于 Rust 的AWS SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，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。

Swift

适用于 Swift 的 SDK

Note

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

将分页与 `describeSecurityGroupsPaginated()` 结合使用。

```
import AWSEC2

/// Return an array of strings giving the names of every security group
/// the user is a member of.
///
/// - Parameter ec2Client: The `EC2Client` to use when calling
///   `describeSecurityGroupsPaginated()`.
///
/// - Returns: An array of strings giving the names of every security
///   group the user is a member of.
func getSecurityGroupNames(ec2Client: EC2Client) async -> [String] {
    let pages = ec2Client.describeSecurityGroupsPaginated(
        input: DescribeSecurityGroupsInput()
    )

    var groupNames: [String] = []

    do {
        for try await page in pages {
            guard let groups = page.securityGroups else {
                print("*** Error: No groups returned.")
                continue
            }

            for group in groups {
                groupNames.append(group.groupName ?? "<unknown>")
            }
        }
    } catch {
        print("*** Error: \(error.localizedDescription)")
    }
}
```

```
    return groupNames
}
```

不分页。

```
import AWSEC2

func describeSecurityGroups(groupId: String) async -> Bool {
    do {
        let output = try await ec2Client.describeSecurityGroups(
            input: DescribeSecurityGroupsInput(
                groupIds: [groupId]
            )
        )

        guard let securityGroups = output.securityGroups else {
            print("No security groups found.")
            return true
        }

        for group in securityGroups {
            print("Group \(group.groupId ?? "<unknown>") found with VPC
\(\group.vpcId ?? "<unknown>")")
        }
        return true
    } catch {
        print("**** Error getting security group details:
\(\error.localizedDescription)")
        return false
    }
}
```

- 有关 API 的详细信息，请参阅适用于 Swift 的 AWS SDK API 参考 [DescribeSecurityGroups](#) 中。

有关 Swift AWS SDK 开发者指南和代码示例的完整列表，请参阅 [使用 AWS SDK 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。

将 `DescribeSnapshotAttribute` 与 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-01234567890abcdef \  
  --attribute createVolumePermission
```

输出：

```
{  
  "SnapshotId": "snap-01234567890abcdef",  
  "CreateVolumePermissions": [  
    {  
      "UserId": "123456789012"  
    }  
  ]  
}
```

有关更多信息，请参阅《Amazon Elastic Compute Cloud 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

适用于 PowerShell V4 的工具

示例 1：此示例描述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 \(V 4\) DescribeSnapshotAttribute](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述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 \(V 5\) 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 SDK 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《[亚马逊 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。该示例使用 --query 参数仅显示快照 IDs 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。该示例使用 --query 参数仅显示快照 IDs。

```

aws ec2 describe-snapshots \
  --filters Name=volume-id,Values=049df61146c4d7901 \
  --query "Snapshots[*].[SnapshotId]" \
  --output text

```

输出：


```
snap-1234567890abcdef0
snap-08637175a712c3fb9
...
```

有关使用筛选条件的其他示例，请参阅 Amazon EC2 用户指南中的[列出和筛选您的资源](#)。

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。

有关使用标签筛选条件的其他示例，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。它仅显示快照 IDs。

```
aws ec2 describe-snapshots \
  --owner-ids 012345678910 \
  --query "Snapshots[?(StartTime<='2020-03-31')].[SnapshotId]"
```

有关使用筛选条件的其他示例，请参阅 Amazon EC2 用户指南中的[列出和筛选您的资源](#)。

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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的快照。

```
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：此示例描述具有“Name”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 \(V 4\) DescribeSnapshots](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的快照。

```
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：此示例描述具有“Name”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 \(V 5\) 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](#) 中查找完整示例，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 SDK 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。

将 `DescribeSpotDatafeedSubscription` 与 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源。

命令:

```
aws ec2 describe-spot-datafeed-subscription
```

输出 :

```
{
  "SpotDatafeedSubscription": {
    "OwnerId": "123456789012",
    "Prefix": "spotdata",
    "Bucket": "amzn-s3-demo-bucket",
    "State": "Active"
  }
}
```

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例描述竞价型实例数据源。

```
Get-EC2SpotDatafeedSubscription
```

输出 :

```
Bucket : amzn-s3-demo-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 \(V 4\) DescribeSpotDatafeedSubscription](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述竞价型实例数据源。

```
Get-EC2SpotDatafeedSubscription
```

输出：

```
Bucket     : amzn-s3-demo-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 \(V 5\) 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 SDK 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。

将 **DescribeSpotFleetInstances** 与 CLI 配合使用

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

CLI

AWS CLI

描述与竞价型实例集关联的竞价型实例

此示例命令列出了与指定竞价型实例集关联的竞价型实例。

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

适用于 PowerShell V4 的工具

示例 1：此示例描述与指定竞价型实例集请求关联的实例。

```
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 (V 4) [DescribeSpotFleetInstances](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述与指定竞价型实例集请求关联的实例。


```
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 \(V 5\) 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 SDK 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。

将 **DescribeSpotFleetRequestHistory** 与 CLI 配合使用

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

CLI

AWS CLI

描述竞价型实例集历史记录

此示例命令返回从指定时间开始的指定竞价型实例集的历史记录。

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实例集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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的竞价型实例集请求的历史记录。

```
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 \(V 4\) DescribeSpotFleetRequestHistory](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的竞价型实例集请求的历史记录。

```
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 \(V 5\) 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 SDK 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。

将 **DescribeSpotFleetRequests** 与 CLI 配合使用

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

CLI

AWS CLI

描述竞价型实例集请求

此示例描述了您的所有竞价型实例集请求。

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"
  }
]
}

```

描述竞价型实例集请求

此示例描述了指定的竞价型实例集请求。

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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的竞价型实例集请求。

```
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：此示例描述所有竞价型实例集请求。

```
Get-EC2SpotFleetRequest
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeSpotFleetRequests](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的竞价型实例集请求。

```
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：此示例描述所有竞价型实例集请求。

```
Get-EC2SpotFleetRequest
```


- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 5\) 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 SDK 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。

将 **DescribeSpotInstanceRequests** 与 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范围限定为在指定可用区中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 IDs。

```
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《Amazon Elastic Compute Cloud 用户指南》中的[列出并筛选您的资源](#)。

示例 3：描述基于标签的竞价型实例请求

以下 `describe-spot-instance-requests` 示例使用标签筛选条件将结果范围限定为具有标签 `cost-center=cc123` 的竞价型实例请求。

```
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。

有关使用标签筛选条件的其他示例，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

适用于 PowerShell V4 的工具

示例 1：此示例描述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 \(V 4\) DescribeSpotInstanceRequests](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述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 \(V 5\) 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 SDK 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。

将 **DescribeSpotPriceHistory** 与 CLI 配合使用

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

CLI

AWS CLI

描述竞价型实例价格历史记录

此示例命令返回 m1.xlarge 实例在 1 月份特定日期的竞价型实例价格历史记录。

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、Amazon Linux/UNIX 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

适用于 PowerShell V4 的工具

示例 1：此示例获取指定实例类型和可用区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了默认区域“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 \(V 4\) DescribeSpotPriceHistory](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例获取指定实例类型和可用区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了默认区域“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 \(V 5\) 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 SDK 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。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 Amazon VPC](#)
- [中转网关入门](#)

.NET

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。

```
/// <summary>
/// Get all the subnets for a Vpc in a set of availability zones.
/// </summary>
/// <param name="vpcId">The Id of the Vpc.</param>
/// <param name="availabilityZones">The list of availability zones.</param>
/// <returns>The collection of subnet objects.</returns>
public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
{
    try
    {
        var subnets = new List<Subnet>();
        var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
            new DescribeSubnetsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
```

```

        {
            new("vpc-id", new List<string>() { vpcId }),
            new("availability-zone", availabilityZones),
            new("default-for-az", new List<string>() { "true" })
        }
    });

    // Get the entire list using the paginator.
    await foreach (var subnet in subnetPaginator.Subnets)
    {
        subnets.Add(subnet);
    }

    return subnets;
}
catch (AmazonEC2Exception ec2Exception)
{
    if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
    {
        _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
    }

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

```

- 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": "111122223333",

```

```

        "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 VPCs 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": "111122223333",
        "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 VPCs 和子网](#)。

示例 3：描述带有特定标签的子网

以下describe-subnets示例使用过滤器检索带有标签的子网的详细信息，CostCenter=123并使用--query参数来显示带有此标签 IDs 的子网的子网。

```

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 VPCs 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](#) 中查找完整示例，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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的子网。

```
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：此示例描述所有子网。

```
Get-EC2Subnet
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeSubnets](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的子网。

```
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：此示例描述所有子网。

```
Get-EC2Subnet
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 5\) DescribeSubnets](#) 中的。

Python

适用于 Python 的 SDK (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。

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

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
```



```

self.ssm_client = ssm_client
self.iam_client = iam_client
sts_client = boto3.client("sts")
self.account_id = sts_client.get_caller_identity()["Account"]

self.key_pair_name = f"{resource_prefix}-key-pair"
self.launch_template_name = f"{resource_prefix}-template-"
self.group_name = f"{resource_prefix}-group"

# Happy path
self.instance_policy_name = f"{resource_prefix}-pol"
self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"

# Failure mode
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
    """
    Gets the default subnets in a VPC for a specified list of Availability
    Zones.

    :param vpc_id: The ID of the VPC to look up.
    :param zones: The list of Availability Zones to look up.
    :return: The list of subnets found.
    """
    # Ensure that 'zones' is a list, even if None is passed
    if zones is None:
        zones = []
    try:
        paginator = self.ec2_client.get_paginator("describe_subnets")
        page_iterator = paginator.paginate(
            Filters=[
                {"Name": "vpc-id", "Values": [vpc_id]},
                {"Name": "availability-zone", "Values": zones},
                {"Name": "default-for-az", "Values": ["true"]},
            ]
        )

        subnets = []

```

```

        for page in page_iterator:
            subnets.extend(page["Subnets"])

        log.info("Found %s subnets for the specified zones.", len(subnets))
        return subnets
    except ClientError as err:
        log.error(
            f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
{zones}."
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidVpcID.NotFound":
            log.error(
                "The specified VPC ID does not exist. "
                "Please check the VPC ID and try again."
            )
        # Add more error-specific handling as needed
        log.error(f"Full error:\n\t{err}")

```

- 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 SDK 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。

将 **DescribeTags** 与 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：描述所有标签

以下 `describe-tags` 示例描述了所有资源的标签。

```
aws ec2 describe-tags
```

示例 4：根据标签键描述资源的标签

以下 `describe-tags` 示例描述了具有键 `Stack` 的标签的资源的标签。

```
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` 示例描述了具有标签 `Stack=Test` 的资源的标签。

```
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具有标签 Stack=Test 的资源。

```
aws ec2 describe-tags \  
--filters "Name=tag:Stack,Values=Test"
```

以下 describe-tags 示例描述了具有键为 Purpose 但没有值的标签的所有实例的标签。

```
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

适用于 PowerShell V4 的工具

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：此示例获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}
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：此示例显示给定区域中所有带有“auto-delete”标签且值为“no”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“Name”键并以表格式显示它们

```

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 (V 4) [DescribeTags](#) 中的。

适用于 PowerShell V5 的工具

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：此示例获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...}
53	instance	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription...}
3	route-table	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription}
5	security-group	{Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription, Amazon.EC2.Model.TagDescription...}


```

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：此示例显示给定区域中所有带有“auto-delete”标签且值为“no”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“Name”键并以表格式显示它们

```

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
```

示例 6：此示例验证使用 DryRun 参数获取 EC2 标签的权限，而无需实际获取标签。注：这在操作成功时会引发异常，这是预期行为。

```
Get-EC2Tag -DryRun $true
```

输出：

```
Get-EC2Tag: Request would have succeeded, but DryRun flag is set.
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) [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 SDK 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。

将 **DescribeVolumeAttribute** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeVolumeAttribute。

CLI

AWS CLI

描述卷属性

此示例命令描述了 ID 为 vol-049df61146c4d7901 的卷的 autoEnableIo 属性。

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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定卷的指定属性。

```
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 (V 4) [DescribeVolumeAttribute](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定卷的指定属性。

```
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 (V 5) [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 SDK 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。

将 **DescribeVolumeStatus** 与 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 `vol-1234567890abcdef0` 的状态。

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，没有受损卷。

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（状态为受损），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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定卷的状态。

```
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 \(V 4\) DescribeVolumeStatus](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定卷的状态。

```
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 \(V 5\) 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 SDK 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。

将 **DescribeVolumes** 与 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` 示例描述了既连接到指定实例又设置为在实例终止时删除的所有卷。

```

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` 且位于指定可用区中的所有卷。

```
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` 示例描述了具有标签键 `Name` 和以 `Test` 开头的值的所有卷。然后，使用仅显示标签和卷的查询对输出 IDs 进行过滤。

```
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"
  }
]
```

有关使用标签筛选条件的其他示例，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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的 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：此示例描述所有 EBS 卷。

```
Get-EC2Volume
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) DescribeVolumes](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的 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：此示例描述所有 EBS 卷。

```
Get-EC2Volume
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 5\) 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 SDK 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。

将 **DescribeVpcAttribute** 与 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 `enableDnsSupport` 属性。此属性指示是否为 VPC 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 enableDnsHostnames 属性。此属性指示在 VPC 中启动的实例是否可获得 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

适用于 PowerShell V4 的工具

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 \(V 4\) DescribeVpcAttribute](#) 中的。

适用于 PowerShell V5 的工具

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 \(V 5\) 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 SDK 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。

将 **DescribeVpcClassicLink** 与 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 ClassicLink 状态 VPCs

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 VPCs 列出已启用 Classiclink 的选项 (的筛选器值设置 `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

适用于 PowerShell V4 的工具

示例 1：上面的示例返回 VPCs 了该区域的所有及其 ClassicLinkEnabled 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 (V 4) [DescribeVpcClassicLink](#)中的。

适用于 PowerShell V5 的工具

示例 1：上面的示例返回 VPCs 了该区域的所有及其 ClassicLinkEnabled 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 (V 5) [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 SDK 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。

将 **DescribeVpcClassicLinkDnsSupport** 与 CLI 配合使用

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

CLI

AWS CLI

描述您 ClassicLink 的 DNS 支持 VPCs

此示例描述了您所有 ClassicLink 的 DNS 支持状态 VPCs。

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

适用于 PowerShell V4 的工具

示例 1：此示例描述了 eu-west-1 区域 VPCs 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 \(V 4\) DescribeVpcClassicLinkDnsSupport](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述了 eu-west-1 区域 VPCs 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 \(V 5\) 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 SDK 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。

将 **DescribeVpcEndpointServices** 与 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 端点服务

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"
]

```

```
}
```

示例 2：描述端点服务的详细信息

以下 `describe-vpc-endpoint-services` 示例列出 Amazon S3 接口端点服务的详细信息。

```
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《AWS PrivateLink 用户指南》中的[查看可用 AWS 服务名称](#)。

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例描述了使用给定筛选条件的 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 (V 4) [DescribeVpcEndpointServices](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述了使用给定筛选条件的 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 \(V 5\) 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 SDK 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。

将 `DescribeVpcEndpoints` 与 CLI 配合使用

以下代码示例演示如何使用 `DescribeVpcEndpoints`。

CLI

AWS CLI

描述 VPC 端点

以下 `describe-vpc-endpoints` 示例显示了所有 VPC 端点的详细信息。

```
aws ec2 describe-vpc-endpoints
```

输出：

```
{
  "VpcEndpoints": [
    {
      "PolicyDocument": "{\n  \"Version\": \"2008-10-17\",\n  \"Statement\": [\n    {\n      \"Effect\": \"Allow\",\n      \"Principal\": \"*\",\n      \"Action\": \"*\",\n      \"Resource\": \"*\n    }\n  ]\n}",
      "VpcId": "vpc-aabb1122",
      "NetworkInterfaceIds": [],
      "SubnetIds": [],
      "PrivateDnsEnabled": true,
      "State": "available",
      "ServiceName": "com.amazonaws.us-east-1.dynamodb",
      "RouteTableIds": [
        "rtb-3d560345"
      ],
      "Groups": [],
      "VpcEndpointId": "vpce-032a826a",
      "VpcEndpointType": "Gateway",
      "CreationTimestamp": "2017-09-05T20:41:28Z",
      "DnsEntries": [],
      "OwnerId": "123456789012"
    },
    {
      "PolicyDocument": "{\n  \"Statement\": [\n    {\n      \"Action\": \"*\n    },\n    {\n      \"Effect\": \"Allow\",\n      \"Principal\": \"*\n    },\n    {\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AWS PrivateLink 用户指南中的[概念](#)。

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例描述区域 eu-west-1 的一个或多个 VPC 端点。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该终端节点

```
Get-EC2VpcEndpoint -Region eu-west-1 | Select-Object VpcEndpointId, VpcId,
ServiceName, PrivateDnsEnabled | Format-Table -AutoSize
```

输出：

VpcEndpointId	VpcId	ServiceName
PrivateDnsEnabled		
-----	-----	-----
vpce-02a2ab2f2f2cc2f2d	vpce-0fc6ff46f65b039eb	com.amazonaws.eu-west-1.ssm
True		
vpce-01d1b111a1114561b	vpce-0fc6ff46f65b039eb	com.amazonaws.eu-west-1.ec2
True		
vpce-0011e23d45167e838	vpce-0fc6ff46f65b039eb	com.amazonaws.eu-west-1.ec2messages
True		
vpce-0c123db4567890123	vpce-0fc6ff46f65b039eb	com.amazonaws.eu-west-1.ssmmessages
True		

示例 3：此示例将 VPC 端点 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 (V 4) [DescribeVpcEndpoints](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述区域 eu-west-1 的一个或多个 VPC 端点。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该终端节点

```
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 端点 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 \(V 5\)](#) [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 SDK 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。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 Amazon VPC](#)
- [中转网关入门](#)
- [VPC IPAM 入门](#)

.NET

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。

```
/// <summary>
/// Get the default VPC for the account.
/// </summary>
/// <returns>The default VPC object.</returns>
public async Task<Vpc> GetDefaultVpc()
{
    try
    {
        var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
            new DescribeVpcsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new("is-default", new List<string>() { "true" })
                }
            });
        return vpcResponse.Vpcs[0];
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "UnauthorizedOperation")
        {
            _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
        }

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

```
        throw;  
    }  
}
```

- 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：描述您的所有内容 VPCs

以下describe-vpcs示例检索有关您的 VPCs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](#) 中查找完整示例，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

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的 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 VPCs 与指定过滤器匹配的（即，具有与值“10.0.0.0/16”匹配且处于“可用”状态的 CIDR）。

```
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内容 VPCs。

```
Get-EC2Vpc
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [DescribeVpcs](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的 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 VPCs 与指定过滤器匹配的（即，具有与值“10.0.0.0/16”匹配且处于“可用”状态的 CIDR）。

```
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内容 VPCs。

```
Get-EC2Vpc
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) [DescribeVpcs](#) 中的。

Python

适用于 Python 的 SDK (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。

```
class AutoScalingWrapper:  
    """  
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.  
    """  
  
    def __init__(  
        self,  
        resource_prefix: str,  
        inst_type: str,  
        ami_param: str,  
        autoscaling_client: boto3.client,  
        ec2_client: boto3.client,  
        ssm_client: boto3.client,  
        iam_client: boto3.client,  
    ):  
        """  
        Initializes the AutoScaler class with the necessary parameters.
```

```

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}])

```

```

    )
except ClientError as err:
    error_code = err.response["Error"]["Code"]
    log.error("Failed to retrieve the default VPC.")
    if error_code == "UnauthorizedOperation":
        log.error(
            "You do not have the necessary permissions to describe VPCs.
            "
            "Ensure that your AWS IAM user or role has the correct
permissions."
        )
    elif error_code == "InvalidParameterValue":
        log.error(
            "One or more parameters are invalid. Check the request
parameters."
        )

    log.error(f"Full error:\n\t{err}")
else:
    if "Vpcs" in response and response["Vpcs"]:
        log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
        return response["Vpcs"][0]
    else:
        pass

```

- 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 SDK 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。

将 **DescribeVpnConnections** 与 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 VPN 连接

以下describe-vpn-connections示例描述了您的所有 Site-to-Site 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《AWS Site-to-Site VPN 用户指南》中的AWS Site-to-Site VPN 工作原理。](#)

示例 2：描述可用的 VPN 连接

以下describe-vpn-connections示例描述了状态为的 Site-to-Site VPN 连接available。

```
aws ec2 describe-vpn-connections \
  --filters "Name=state,Values=available"
```

有关更多信息，[请参阅《AWS Site-to-Site VPN 用户指南》中的AWS Site-to-Site VPN 工作原理。](#)

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例描述指定的 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：此示例描述状态为待定或可用的任何 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 (V 4) [DescribeVpnConnections](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述指定的 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：此示例描述状态为待定或可用的任何 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 \(V 5\) 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 SDK 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。

将 **DescribeVpnGateways** 与 CLI 配合使用

以下代码示例演示如何使用 DescribeVpnGateways。

CLI

AWS CLI

描述虚拟私有网关

此示例描述了您的虚拟私有网关。

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

适用于 PowerShell V4 的工具

示例 1：此示例描述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 \(V 4\) DescribeVpnGateways](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例描述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 \(V 5\) 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 SDK 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。

将 **DetachInternetGateway** 与 CLI 配合使用

以下代码示例演示如何使用 DetachInternetGateway。

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：

- [创建具有私有子网和 NAT 网关的 VPC](#)
- [开始使用 Amazon VPC](#)

CLI

AWS CLI

将互联网网关与 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

适用于 PowerShell V4 的工具

示例 1：此示例将指定的互联网网关与指定的 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 (V 4) [DetachInternetGateway](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例将指定的互联网网关与指定的 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 \(V 5\) 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 SDK 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。

将 **DetachNetworkInterface** 与 CLI 配合使用

以下代码示例演示如何使用 DetachNetworkInterface。

CLI

AWS CLI

将网络接口与实例分离

此示例将指定的网络接口与指定的实例分离。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

适用于 PowerShell V4 的工具

示例 1：此示例移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 \(V 4\) DetachNetworkInterface](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例移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 \(V 5\) 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 SDK 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。

将 **DetachVolume** 与 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) 与其连接到的实例分离。

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

适用于 PowerShell V4 的工具

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 \(V 4\)](#) [DetachVolume](#) 中的。

适用于 PowerShell V5 的工具

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 \(V 5\)](#) [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 SDK 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。

将 **DetachVpnGateway** 与 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 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

适用于 PowerShell V4 的工具

示例 1：此示例将指定的虚拟专用网关与指定的 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 (V 4) [DetachVpnGateway](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例将指定的虚拟专用网关与指定的 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 (V 5) [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 SDK 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。

将 **DisableVgwRoutePropagation** 与 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私有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 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

适用于 PowerShell V4 的工具

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 (V 4) [DisableVgwRoutePropagation](#)中的。

适用于 PowerShell V5 的工具

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 (V 5) [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 SDK 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。

将 **DisableVpcClassicLink** 与 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-88888888 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

适用于 PowerShell V4 的工具

示例 1：此示例禁用 vpc-01e23c4a5d6d EC2 VpcClassicLink b78e9。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 (V 4) [DisableVpcClassicLink](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例禁用 vpc-01e23c4a5d6d EC2 VpcClassicLink b78e9。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 \(V 5\) 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 SDK 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。

将 **DisableVpcClassicLinkDnsSupport** 与 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

适用于 PowerShell V4 的工具

示例 1：此示例禁用了 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 \(V 4\) DisableVpcClassicLinkDnsSupport](#) 中的。

适用于 PowerShell V5 的工具

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 \(V 5\) 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 SDK 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：

- [了解基本功能](#)

.NET

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。

```
/// <summary>  
/// Disassociate an Elastic IP address from an EC2 instance.  
/// </summary>  
/// <param name="associationId">The association Id.</param>  
/// <returns>A Boolean value indicating the success of the action.</returns>  
public async Task<bool> DisassociateIp(string associationId)  
{
```

```

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

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

```

- 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](#) 中查找完整示例，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

在-Classic 中取消弹性 IP 地址的 EC2关联

此示例在 EC2-Classic 中取消弹性 IP 地址与实例的关联。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
```

取消与-VPC 中的弹性 IP 地址的 EC2关联

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](#) 中查找完整示例，了解如何进行设置和运行。

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

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

    return response;
}
```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

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

/**
 * Disassociate an Elastic IP address from an instance.
 * @param {{ associationId: string }} options
 */
export const main = async ({ associationId }) => {
  const client = new EC2Client({});
  const command = new DisassociateAddressCommand({
    // You can also use PublicIp, but that is for EC2 classic which is being
    retired.
    AssociationId: associationId,
  });

  try {
    await client.send(command);
    console.log("Successfully disassociated address");
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidAssociationID.NotFound"
    ) {
      console.warn(`${caught.message}.`);
    } else {
      throw caught;
    }
  }
};
```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```
suspend fun disassociateAddressSc(associationIdVal: String?) {  
    val addressRequest =  
        DisassociateAddressRequest {  
            associationId = associationIdVal  
        }  
    Ec2Client.fromEnvironment { 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 Kotlin 的 AWS SDK API 参考。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将指定的弹性 IP 地址与 VPC 中的指定实例取消关联。

```
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 \(V 4\) DisassociateAddress](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例将指定的弹性 IP 地址与 VPC 中的指定实例取消关联。

```
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 \(V 5\) DisassociateAddress](#) 中的。

Python

适用于 Python 的 SDK (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。

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

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
            = None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
```

```

        :param public_ip: The public IP address of the Elastic IP.
        :param instance_id: The ID of the associated EC2 instance, if any.
        """
        self.allocation_id = allocation_id
        self.public_ip = public_ip
        self.instance_id = instance_id

def __init__(self, ec2_client: Any) -> None:
    """
    Initializes the ElasticIpWrapper with an EC2 client.

    :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                        access to AWS EC2 services.
    """
    self.ec2_client = ec2_client
    self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

    @classmethod
    def from_client(cls) -> "ElasticIpWrapper":
        """
        Creates an ElasticIpWrapper instance with a default EC2 client.

        :return: An instance of ElasticIpWrapper initialized with the default EC2
client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def disassociate(self, allocation_id: str) -> None:
        """
        Removes an association between an Elastic IP address and an instance.
When the
        association is removed, the instance is assigned a new public IP address.

        :param allocation_id: The allocation ID of the Elastic IP to
disassociate.
        :raises ClientError: If the disassociation fails, such as when the
association ID is not found.
        """
        elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
allocation_id)
        if elastic_ip is None or elastic_ip.instance_id is None:

```

```

        logger.info(
            f"No association found for Elastic IP with allocation ID
{allocation_id}."
        )
        return

    try:
        # Retrieve the association ID before disassociating
        response =
self.ec2_client.describe_addresses(AllocationIds=[allocation_id])
        association_id = response["Addresses"][0].get("AssociationId")

        if association_id:

self.ec2_client.disassociate_address(AssociationId=association_id)
            elastic_ip.instance_id = None # Remove the instance association
        else:
            logger.info(
                f"No Association ID found for Elastic IP with allocation ID
{allocation_id}."
            )

    except ClientError as err:
        if err.response["Error"]["Code"] == "InvalidAssociationID.NotFound":
            logger.error(
                f"Failed to disassociate Elastic IP {allocation_id} "
                "because the specified association ID for the Elastic IP
address was not found. "
                "Verify the association ID and ensure the Elastic IP is
currently associated with a "
                "resource before attempting to disassociate it."
            )
        raise

```

- 有关 API 的详细信息，请参阅适用[DisassociateAddress](#)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](#) 中查找完整示例，了解如何进行设置和运行。

```
pub async fn disassociate_ip_address(&self, association_id: &str) ->
Result<(), EC2Error> {
    self.client
        .disassociate_address()
        .association_id(association_id)
        .send()
        .await?;
    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用[DisassociateAddress](#)于 Rust 的AWS SDK API 参考。

Swift

适用于 Swift 的 SDK

Note

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

```
import AWSEC2

/// Disassociate an Elastic IP.
///
/// - Parameter associationId: The ID of the association to end.
func disassociateAddress(associationId: String?) async {
    do {
```

```
        _ = try await ec2Client.disassociateAddress(  
            input: DisassociateAddressInput(  
                associationId: associationId  
            )  
        )  
    } catch {  
        print("*** Unable to disassociate the IP address:  
\\(error.localizedDescription)")  
    }  
}
```

- 有关 API 的详细信息，请参阅适用于 Swift 的 AWS SDK API 参考[DisassociateAddress](#)中。

有关 AWS SDK 开发者指南和代码示例的完整列表，请参阅[使用 AWS SDK 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。

将 **DisassociateRouteTable** 与 CLI 配合使用

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

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 Amazon VPC](#)

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

适用于 PowerShell V4 的工具

示例 1：此示例移除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 \(V 4\) DisassociateRouteTable](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例移除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 \(V 5\) 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 SDK 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。

将 **EnableVgwRoutePropagation** 与 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私有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

适用于 PowerShell V4 的工具

示例 1：此示例支持指定的 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 \(V 4\) EnableVgwRoutePropagation](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例支持指定的 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 \(V 5\) 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 SDK 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。

将 **EnableVolumeIo** 与 CLI 配合使用

以下代码示例演示如何使用 EnableVolumeIo。

CLI

AWS CLI

I/O 为卷启用

此示例 I/O 在音量上启用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

适用于 PowerShell V4 的工具

示例 1：如果禁用了 I/O 操作，则此示例将启用对指定卷的 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 (V 4) [EnableVolumeIO](#)中的。

适用于 PowerShell V5 的工具

示例 1：如果禁用了 I/O 操作，则此示例将启用对指定卷的 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 (V 5) [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 SDK 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。

将 **EnableVpcClassicLink** 与 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-88888888。 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

适用于 PowerShell V4 的工具

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 (V 4) [EnableVpcClassicLink](#)中的。

适用于 PowerShell V5 的工具

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 (V 5) [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 SDK 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。

将 **EnableVpcClassicLinkDnsSupport** 与 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

此示例为启用了 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

适用于 PowerShell V4 的工具

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 (V 4) [EnableVpcClassicLinkDnsSupport](#) 中的。

适用于 PowerShell V5 的工具

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 \(V 5\) 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 SDK 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。

将 **GetConsoleOutput** 与 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

适用于 PowerShell V4 的工具

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：此示例将经过编码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 (V 4) [GetConsoleOutput](#)中的。

适用于 PowerShell V5 的工具

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：此示例将经过编码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 \(V 5\)](#) [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 SDK 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。

将 `GetHostReservationPurchasePreview` 与 CLI 配合使用

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

CLI

AWS CLI

获取专属主机预留的购买预览

此示例提供您账户中指定专属主机的指定专属主机预留成本的预览。

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

适用于 PowerShell V4 的工具

示例 1：此示例预览配置与专属主机 h-01e23f4cd567890f1 的配置匹配的预留购买。

```
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 (V 4) [GetHostReservationPurchasePreview](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例预览配置与专属主机 h-01e23f4cd567890f1 的配置匹配的预留购买。

```
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 \(V 5\) 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 SDK 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

获取加密密码

此示例将获取加密密码。

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+ga0uN0enX1MmgXPTj7XEXAMPLE
UQ+YeFfb+L1U4C4AKv652Ux1iRB3CPTYP7WmU3TUnhsuBd+p6LVk7T2lKUm160Xbk6WPW1VYYm/TRPB1
e1DQ7PY4an/DgZT4mwcpRfigzhniQgDDe01InvSDcwoUTwNs0Y1S8ouri2W4n5GNlriM3Q0AnNVelVz/
53TkDtxbNoU606M1gK9zUWSxqEgwvbV2j8c5rP0WCuaMWSF14ziDu4bd7q+4RSyi8NUsVWnKZ4aEZffu
DPGzKrF5yLlf3etP2L4ZR6CvG7K1hx7VK0QVN32Dajw=="
}
```

获取解密密码

此示例将获取解密密码。

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。

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。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.ec2.Ec2AsyncClient;
import software.amazon.awssdk.services.ec2.model.*;
import java.util.concurrent.CompletableFuture;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
```

```

public class GetPasswordData {

    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <instanceId>

            Where:
                instanceId - An instance id value that you can obtain from the
AWS Management Console.\s
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        String instanceId = args[0];
        Ec2AsyncClient ec2AsyncClient = Ec2AsyncClient.builder()
            .region(Region.US_EAST_1)
            .build();

        try {
            CompletableFuture<Void> future = getPasswordDataAsync(ec2AsyncClient,
instanceId);
            future.join();
        } catch (RuntimeException rte) {
            System.err.println("An exception occurred: " + (rte.getCause() !=
null ? rte.getCause().getMessage() : rte.getMessage()));
        }
    }

    /**
     * Fetches the password data for the specified EC2 instance asynchronously.
     *
     * @param ec2AsyncClient the EC2 asynchronous client to use for the request
     * @param instanceId instanceId the ID of the EC2 instance for which you want
to fetch the password data
     * @return a {@link CompletableFuture} that completes when the password data
has been fetched
     * @throws RuntimeException if there was a failure in fetching the password
data
     */
}

```

```
public static CompletableFuture<Void> getPasswordDataAsync(Ec2AsyncClient
ec2AsyncClient, String instanceId) {
    GetPasswordDataRequest getPasswordDataRequest =
GetPasswordDataRequest.builder()
        .instanceId(instanceId)
        .build();

    CompletableFuture<GetPasswordDataResponse> response =
ec2AsyncClient.getPasswordData(getPasswordDataRequest);
    response.whenComplete((getPasswordDataResponse, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to get password data for
instance: " + instanceId, ex);
        } else if (getPasswordDataResponse == null ||
getPasswordDataResponse.getPasswordData().isEmpty()) {
            throw new RuntimeException("No password data found for instance:
" + instanceId);
        } else {
            String encryptedPasswordData =
getPasswordDataResponse.getPasswordData();
            System.out.println("Encrypted Password Data: " +
encryptedPasswordData);
        }
    });

    return response.thenApply(resp -> null);
}
}
```

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例解密亚马逊为指定 Windows 实例的管理员账户 EC2 分配的密码。当指定了 pem 文件时，将自动假定 -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，则对密码进行解密。

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

输出：

```
mYZ(PA9?C)Q
```

示例 3：返回实例的已加密密码数据。

```
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 \(V 4\)](#) [GetPasswordData](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例解密亚马逊为指定 Windows 实例的管理员账户 EC2 分配的密码。当指定了 pem 文件时，将自动假定 -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，则对密码进行解密。

```
Get-EC2PasswordData -InstanceId i-12345678 -Decrypt
```

输出：

```
mYZ(PA9?C)Q
```

示例 3：返回实例的已加密密码数据。

```
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 \(V 5\)](#) [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 SDK 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。

将 **ImportImage** 与 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

适用于 PowerShell V4 的工具

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 EC2 Amazon。该示例要求存在默认名称为“vmimport”的虚拟机导入服务角色，其策略允许亚马逊 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 = "amzn-s3-demo-bucket"
$container.UserBucket.S3Key = "Win_2008_Server_Standard_SP2_64-bit-disk1.vmdk"

$params = @{
    "ClientToken"="idempotencyToken"
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
    "LicenseType"="AWS"
}

Import-EC2Image -DiskContainer $container @params

```

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 (V 4) [ImportImage](#) 中的。

适用于 PowerShell V5 的工具

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 EC2 Amazon。该示例要求存在默认名称为“vmimport”的虚拟机导入服务角色，其策略允许亚马逊 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 = "amzn-s3-demo-bucket"
$container.UserBucket.S3Key = "Win_2008_Server_Standard_SP2_64-bit-disk1.vmdk"

$params = @{
    "ClientToken"="idempotencyToken"
    "Description"="Windows 2008 Standard Image Import"
    "Platform"="Windows"
    "LicenseType"="AWS"
}

Import-EC2Image -DiskContainer $container @params

```

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 (V 5) [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 SDK 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。

将 **ImportKeyPair** 与 CLI 配合使用

以下代码示例演示如何使用 `ImportKeyPair`。

CLI

AWS CLI

导入公有密钥

首先，使用您选择的工具生成密钥对。例如，使用此 `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有密钥。

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

适用于 PowerShell V4 的工具

示例 1：此示例将公钥导入到 EC2。第一行将公有密钥文件 (*.pub) 的内容存储在变量 **\$publickey** 中。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 (V 4) [ImportKeyPair](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例将公钥导入到 EC2。第一行将公有密钥文件 (*.pub) 的内容存储在变量 **\$publickey** 中。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 (V 5) [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 SDK 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。

将 **ImportSnapshot** 与 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

适用于 PowerShell V4 的工具

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/AWSEC2/latest/WindowsGuide/VMImportPrerequisites.html>。要使用自定义角色，请使用 **-RoleName** 参数指定角色名称。

```

$params = @{
    "ClientToken"="idempotencyToken"
    "Description"="Disk Image Import"
    "DiskContainer_Description" = "Data disk"
    "DiskContainer_Format" = "VMDK"
    "DiskContainer_S3Bucket" = "amzn-s3-demo-bucket"
    "DiskContainer_S3Key" = "datadiskimage.vmdk"
}

Import-EC2Snapshot @params

```

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 \(V 4\)](#) [ImportSnapshot](#) 中的。

适用于 PowerShell V5 的工具

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/AWSEC2/latest/WindowsGuide/VMImportPrerequisites.html>。要使用自定义角色，请使用 **-RoleName** 参数指定角色名称。

```

$params = @{
    "ClientToken"="idempotencyToken"
    "Description"="Disk Image Import"
    "DiskContainer_Description" = "Data disk"
    "DiskContainer_Format" = "VMDK"
    "DiskContainer_S3Bucket" = "amzn-s3-demo-bucket"
    "DiskContainer_S3Key" = "datadiskimage.vmdk"
}

Import-EC2Snapshot @params

```

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 \(V 5\)](#) [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 SDK 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。

将 **ModifyCapacityReservation** 与 CLI 配合使用

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

CLI

AWS CLI

示例 1：更改现有容量预留所预留的实例数量

以下 `modify-capacity-reservation` 示例将更改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  
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[修改容量预留](#)。

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 Amazon EC2 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

适用于 PowerShell V4 的工具

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 \(V 4\) `ModifyCapacityReservation`](#) 中的。

适用于 PowerShell V5 的工具

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 \(V 5\) `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 SDK 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。

将 **ModifyHosts** 与 CLI 配合使用

以下代码示例演示如何使用 ModifyHosts。

CLI

AWS CLI

示例 1：为专属主机启用自动置放

以下 modify-hosts 示例为专属主机启用自动置放，以便它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": []  
}
```

有关更多信息，请参阅 Amazon EC2 用户指南中的[修改专用主机的自动放置设置](#)。

示例 2：为专属主机启用主机恢复

以下 modify-hosts 示例为指定的专属主机启用主机恢复。

```
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 Amazon EC2 用户指南中的[修改专用主机的自动放置设置](#)。

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

PowerShell

适用于 PowerShell V4 的工具

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 (V 4) [ModifyHosts](#) 中的。

适用于 PowerShell V5 的工具

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 (V 5) [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 SDK 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。

将 **ModifyIdFormat** 与 CLI 配合使用

以下代码示例演示如何使用 `ModifyIdFormat`。

CLI

AWS CLI

为资源启用较长 ID 格式

以下 `modify-id-format` 示例为 `instance` 资源类型启用较长 ID 格式。

```
aws ec2 modify-id-format \  
  --resource instance \  
  --use-long-ids
```

为资源禁用较长 ID 格式

以下 modify-id-format 示例为 instance 资源类型禁用较长 ID 格式。

```
aws ec2 modify-id-format \  
  --resource instance \  
  --no-use-long-ids
```

以下 modify-id-format 示例为处于选择周期内的所有支持的资源类型启用较长 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

适用于 PowerShell V4 的工具

示例 1：此示例为指定的资源类型启用较长的 ID 格式。

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

示例 2：此示例为指定的资源类型禁用较长的 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 (V 4) [ModifyIdFormat](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例为指定的资源类型启用较长的 ID 格式。

```
Edit-EC2IdFormat -Resource instance -UseLongId $true
```

示例 2：此示例为指定的资源类型禁用较长的 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 \(V 5\) 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 SDK 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。

将 **ModifyImageAttribute** 与 CLI 配合使用

以下代码示例演示如何使用 `ModifyImageAttribute`。

CLI

AWS CLI

示例 1：公开 AMI

以下 `modify-instance-attribute` 示例将公开指定的 AMI。

```
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

适用于 PowerShell V4 的工具

示例 1：此示例更新指定 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 \(V 4\) `ModifyImageAttribute`](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例更新指定 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 \(V 5\) `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 SDK 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。

将 **ModifyInstanceAttribute** 与 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`。该实例必须在 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` 示例将指定的 Amazon EBS 支持的实例的根卷的 `deleteOnTermination` 属性设置为 `false`。默认情况下，根卷的此属性为 `true`。

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 `UserData.txt` 的内容：

```
#!/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。

此命令的 Linux/macOS 版本：

```
base64 UserData.txt > UserData.base64.txt
```

此命令不生成任何输出。

此命令的 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，您可以在以下 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[用户指南中的用户数据和 AWS CLI](#)。EC2

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例修改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“simple”指定为单根虚拟化 (SR-IOV) 网络支持参数的值，为指定实例启用增强联网，方法是将“simple”指定为单根 I/O 虚拟化 (SR-IOV) 网络支持参数的值，...

SriovNetSupport

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

示例 3：此示例修改指定实例的安全组。该实例必须在 VPC 中。您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。此功能并非适用于所有实例类型。使用 EBS 优化的实例时，可能需要支付额外的使用费。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

示例 5：此示例启用了对指定实例的 source/destination 检查。对于要执行 NAT 的 NAT 实例，该值必须为“false”。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

示例 6：此示例对指定的实例禁用终止。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

示例 7：此示例更改指定的实例，使其在从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 \(V 4\) ModifyInstanceAttribute](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例修改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“simple”指定为单根虚拟化 (SR-IOV) 网络支持参数的值，为指定实例启用增强联网，方法是将“simple”指定为单根 I/O 虚拟化 (SR-IOV) 网络支持参数的值，--SriovNetSupport

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SriovNetSupport "simple"
```

示例 3：此示例修改指定实例的安全组。该实例必须在 VPC 中。您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。此功能并非适用于所有实例类型。使用 EBS 优化的实例时，可能需要支付额外的使用费。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -EbsOptimized $true
```

示例 5：此示例启用了对指定实例的 source/destination 检查。对于要执行 NAT 的 NAT 实例，该值必须为“false”。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -SourceDestCheck $true
```

示例 6：此示例对指定的实例禁用终止。

```
Edit-EC2InstanceAttribute -InstanceId i-12345678 -DisableApiTermination $true
```

示例 7：此示例更改指定的实例，使其在从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 \(V 5\) 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 SDK 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。

将 **ModifyInstanceCreditSpecification** 与 CLI 配合使用

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

CLI

AWS CLI

修改实例的 CPU 使用率积分选项

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”。有效的积分选项为“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

适用于 PowerShell V4 的工具

示例 1：这会为实例 i-01234567890abcdef 启用 T2 无限积分。

```
$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 (V 4) [ModifyInstanceCreditSpecification](#)中的。

适用于 PowerShell V5 的工具

示例 1：这会为实例 i-01234567890abcdef 启用 T2 无限积分。

```
$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 (V 5) [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 SDK 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。

将 `ModifyNetworkInterfaceAttribute` 与 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

适用于 PowerShell V4 的工具

示例 1：此示例修改指定的网络接口，以便在终止时删除指定的连接。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

示例 2：此示例修改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：此示例禁用对指定网络接口的 source/destination 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 (V 4) [ModifyNetworkInterfaceAttribute](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例修改指定的网络接口，以便在终止时删除指定的连接。

```
Edit-EC2NetworkInterfaceAttribute -NetworkInterfaceId eni-1a2b3c4d -Attachment_AttachmentId eni-attach-1a2b3c4d -Attachment_DeleteOnTermination $true
```

示例 2：此示例修改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：此示例禁用对指定网络接口的 source/destination 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 \(V 5\) 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 SDK 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。

将 **ModifyReservedInstances** 与 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区域的另一个可用区。

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-Classical 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 实例的预留 Linux/UNIX 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 变为 1 个 m1.medium 实例，位于同一可用区中。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-Classical,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

适用于 PowerShell V4 的工具

示例 1：此示例修改指定预留实例的可用区、实例计数和平台。

```
$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 (V 4) [ModifyReservedInstances](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例修改指定预留实例的可用区、实例计数和平台。

```
$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 (V 5) [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 SDK 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。

将 **ModifySnapshotAttribute** 与 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` 示例将公开指定的快照。

```
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

适用于 PowerShell V4 的工具

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 (V 4) [ModifySnapshotAttribute](#) 中的。

适用于 PowerShell V5 的工具

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 \(V 5\)](#) [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 SDK 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。

将 `ModifySpotFleetRequest` 与 CLI 配合使用

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

CLI

AWS CLI

修改竞价型实例集请求

此示例命令更新指定竞价型实例集请求的目标容量。

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  
}
```

此示例命令会减少指定竞价型实例集请求的目标容量，而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

适用于 PowerShell V4 的工具

示例 1：此示例更新指定竞价型实例集请求的目标容量。

```
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 (V 4) [ModifySpotFleetRequest](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例更新指定竞价型实例集请求的目标容量。

```
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 \(V 5\) `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 SDK 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。

将 `ModifySubnetAttribute` 与 CLI 配合使用

以下代码示例演示如何使用 `ModifySubnetAttribute`。

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 Amazon VPC](#)

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地址。IPv4 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`，以指定向该子网启动的所有实例都分配了该子网范围内的地址。IPv6

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 Virtual Private Cloud 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

适用于 PowerShell V4 的工具

示例 1：此示例为指定的子网启用公有 IP 寻址。

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

示例 2：此示例为指定的子网禁用公有 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 \(V 4\)](#) [ModifySubnetAttribute](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例为指定的子网启用公有 IP 寻址。

```
Edit-EC2SubnetAttribute -SubnetId subnet-1a2b3c4d -MapPublicIpOnLaunch $true
```

示例 2：此示例为指定的子网禁用公有 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 \(V 5\)](#) [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 SDK 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。

将 **ModifyVolumeAttribute** 与 CLI 配合使用

以下代码示例演示如何使用 `ModifyVolumeAttribute`。

CLI

AWS CLI

修改卷属性

此示例将 ID 为 `vol-1234567890abcdef0` 的卷的 `autoEnableIo` 属性设置为 `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

适用于 PowerShell V4 的工具

示例 1：此示例修改了指定卷的指定属性。I/O 由于数据可能不一致，该卷的操作将在暂停后自动恢复。

```
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 (V 4) [ModifyVolumeAttribute](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例修改了指定卷的指定属性。I/O 由于数据可能不一致，该卷的操作将在暂停后自动恢复。

```
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 (V 5) [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 SDK 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。

将 **ModifyVpcAttribute** 与 CLI 配合使用

以下代码示例演示如何使用 `ModifyVpcAttribute`。

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 Amazon VPC](#)

CLI

AWS CLI

修改该 enableDnsSupport 属性

此示例修改 enableDnsSupport 属性。此属性指示是否为 VPC 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}"
```

修改该 enableDnsHostnames 属性

此示例修改 enableDnsHostnames 属性。此属性指示在 VPC 中启动的实例是否可获得 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

适用于 PowerShell V4 的工具

示例 1：此示例启用对指定 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：此示例启用对指定 VPC 的 DNS 解析的支持。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

示例 4：此示例禁用对指定 VPC 的 DNS 解析的支持。

```
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 (V 4) [ModifyVpcAttribute](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例启用对指定 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：此示例启用对指定 VPC 的 DNS 解析的支持。

```
Edit-EC2VpcAttribute -VpcId vpc-12345678 -EnableDnsSupport $true
```

示例 4：此示例禁用对指定 VPC 的 DNS 解析的支持。

```
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 (V 5) [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 SDK 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](#) 中查找完整示例，了解如何进行设置和运行。

```

//! Enable detailed monitoring for an Amazon Elastic Compute Cloud (Amazon EC2)
instance.
/*!
\param instanceId: An EC2 instance ID.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::EC2::enableMonitoring(const Aws::String &instanceId,
                                   const Aws::Client::ClientConfiguration
                                   &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::MonitorInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::MonitorInstancesOutcome dryRunOutcome =
    ec2Client.MonitorInstances(request);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to enable monitoring on instance. A dry run
            should trigger an error."
            <<
            std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType()
               != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cerr << "Failed dry run to enable monitoring on instance " <<
            instanceId << ": " << dryRunOutcome.GetError().GetMessage() <<

```

```
        std::endl;
    return false;
}

request.SetDryRun(false);
Aws::EC2::Model::MonitorInstancesOutcome monitorInstancesOutcome =
ec2Client.MonitorInstances(request);
if (!monitorInstancesOutcome.IsSuccess()) {
    std::cerr << "Failed to enable monitoring on instance " <<
        instanceId << ": " <<
        monitorInstancesOutcome.GetError().GetMessage() << std::endl;
} else {
    std::cout << "Successfully enabled monitoring on instance " <<
        instanceId << std::endl;
}

return monitorInstancesOutcome.IsSuccess();
}
```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

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

/**
 * Turn on detailed monitoring for the selected instance.
 * By default, metrics are sent to Amazon CloudWatch every 5 minutes.
 * For a cost you can enable detailed monitoring which sends metrics every
 * minute.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new MonitorInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { InstanceMonitorings } = await client.send(command);
    const instancesBeingMonitored = InstanceMonitorings.map(
      (im) =>
        ` • Detailed monitoring state for ${im.InstanceId} is
        ${im.Monitoring.State}.`,
    );
    console.log("Monitoring status:");
    console.log(instancesBeingMonitored.join("\n"));
  } catch (caught) {
    if (caught instanceof Error && caught.name === "InvalidParameterValue") {
```

```
        console.warn(`${caught.message}`);
    } else {
        throw caught;
    }
}
};
```

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例启用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 (V 4) [MonitorInstances](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例启用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 (V 5) [MonitorInstances](#)中的。

SAP ABAP

适用于 SAP ABAP 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，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 SDK 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。

将 **MoveAddressToVpc** 与 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-VPC 平台。EC2

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

适用于 PowerShell V4 的工具

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）区域的 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 \(V 4\) MoveAddressToVpc](#) 中的。

适用于 PowerShell V5 的工具

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）区域的 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 \(V 5\) 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 SDK 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。

将 **PurchaseHostReservation** 与 CLI 配合使用

以下代码示例演示如何使用 PurchaseHostReservation。

CLI

AWS CLI

购买专属主机预留

此示例为您的账户中的指定专属主机购买指定的专属主机预留产品。

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

适用于 PowerShell V4 的工具

示例 1：此示例购买可提供 0c1f23456789d0ab 的预留，其配置与专属主机 h-01e23f4cd567890f1 的配置匹配

```
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 \(V 4\) PurchaseHostReservation](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例购买可提供 0c1f23456789d0ab 的预留，其配置与专属主机 h-01e23f4cd567890f1 的配置匹配

```
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 \(V 5\) 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 SDK 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。

将 **PurchaseScheduledInstances** 与 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

此示例将购买计划实例。

命令:

```
aws ec2 purchase-scheduled-instances --purchase-requests file://purchase-request.json
```

Purchase-request.json :

```
[
  {
    "PurchaseToken": "eyJ2IjoiMSIsInMiOjEsImMiOi...",
    "InstanceCount": 1
  }
]
```

输出 :

```
{
  "ScheduledInstanceSet": [
    {
      "AvailabilityZone": "us-west-2b",
      "ScheduledInstanceId": "sci-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

适用于 PowerShell V4 的工具

示例 1：此示例购买计划实例。

```
$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-1234-1234-1234-123456789012
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 (V 4) [PurchaseScheduledInstances](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例购买计划实例。

```
$request = New-Object Amazon.EC2.Model.PurchaseRequest
$request.InstanceCount = 1
$request.PurchaseToken = "eyJ2IjoiMSIsInMiOiEsImMiOi..."
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 \(V 5\) 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 SDK 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

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。

按实例的 ID 重启实例。

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

        await _amazonEC2.RebootInstancesAsync(request);

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

        return true;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidInstanceId")
        {
            _logger.LogError(
                $"InstanceId {ec2InstanceId} is invalid, unable to reboot.
{ec2Exception.Message}");
        }
    }
}
```

```

        return false;
    }
    catch (Exception ex)
    {
        _logger.LogError(
            $"An error occurred while rebooting the instance {ec2InstanceId}.: {ex.Message}");
        return false;
    }
}

/// <summary>
/// Wait until an EC2 instance is in a specified state.
/// </summary>
/// <param name="instanceId">The instance Id.</param>
/// <param name="stateName">The state to wait for.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
{
    var request = new DescribeInstancesRequest
    {
        InstanceIds = new List<string> { instanceId }
    };

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

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

    return hasState;
}

```

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)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
            }
        }
    }

```



```

        await foreach (var instance in
instancesPaginator.InstanceInformationList)
        {
            instanceReady = instance.InstanceId == instanceId;
            if (instanceReady)
            {
                break;
            }
        }
    }
    Console.WriteLine("Waiting for instance to be running.");
    await WaitForInstanceState(instanceId, InstanceStateName.Running);
    Console.WriteLine("Instance ready.");
    Console.WriteLine($"Sending restart command to instance
{instanceId}");
    await _amazonSsm.SendCommandAsync(
        new SendCommandRequest()
        {
            InstanceIds = new List<string>() { instanceId },
            DocumentName = "AWS-RunShellScript",
            Parameters = new Dictionary<string, List<string>>()
            {
                {
                    "commands",
                    new List<string>() { "cd / && sudo python3 server.py
80" }
                }
            }
        });
    Console.WriteLine($"Restarted the web server on instance
{instanceId}");
}
catch (AmazonEC2Exception ec2Exception)
{
    if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
    {
        _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
    }

    throw;
}
catch (Exception ex)
{

```

```

        _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
        throw;
    }
}

```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考[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](#) 中查找完整示例，了解如何进行设置和运行。

```

//! Reboot an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
    \param instanceID: An EC2 instance ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::rebootInstance(const Aws::String &instanceId,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::RebootInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::RebootInstancesOutcome dry_run_outcome =
ec2Client.RebootInstances(request);
    if (dry_run_outcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to reboot on instance. A dry run should
trigger an error."
            <<
            std::endl;
    }
}

```

```
        return false;
    } else if (dry_run_outcome.GetError().GetErrorType()
               != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to reboot instance " << instanceId << ": "
                   << dry_run_outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::RebootInstancesOutcome outcome =
ec2Client.RebootInstances(request);
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to reboot instance " << instanceId << ": " <<
                   outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully rebooted instance " << instanceId <<
                   std::endl;
    }

    return outcome.IsSuccess();
}
```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

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

/**
 * Requests a reboot of the specified instances. This operation is asynchronous;
 * it only queues a request to reboot the specified instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new RebootInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    await client.send(command);
    console.log("Instance rebooted successfully.");
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(
        `${caught.message}. Please provide the InstanceId of a valid instance to reboot.`
      );
    } else {
      throw caught;
    }
  }
};
```

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例重启指定的实例。

```
Restart-EC2Instance -InstanceId i-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [RebootInstances](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例重启指定的实例。

```
Restart-EC2Instance -InstanceId i-12345678
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) [RebootInstances](#) 中的。

Python

适用于 Python 的 SDK (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。

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

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
```

```

        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

    def replace_instance_profile(

```

```

        self,
        instance_id: str,
        new_instance_profile_name: str,
        profile_association_id: str,
    ) -> None:
        """
        Replaces the profile associated with a running instance. After the
        profile is
        replaced, the instance is rebooted to ensure that it uses the new
        profile. When
        the instance is ready, Systems Manager is used to restart the Python web
        server.

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
        associate with
                                   the specified instance.
        :param profile_association_id: The ID of the existing profile association
        for the
                                   instance.
        """
        try:
            self.ec2_client.replace_iam_instance_profile_association(
                IamInstanceProfile={"Name": new_instance_profile_name},
                AssociationId=profile_association_id,
            )
            log.info(
                "Replaced instance profile for association %s with profile %s.",
                profile_association_id,
                new_instance_profile_name,
            )
            time.sleep(5)

            self.ec2_client.reboot_instances(InstanceIds=[instance_id])
            log.info("Rebooting instance %s.", instance_id)
            waiter = self.ec2_client.get_waiter("instance_running")
            log.info("Waiting for instance %s to be running.", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info("Instance %s is now running.", instance_id)

            self.ssm_client.send_command(
                InstanceIds=[instance_id],
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )

```

```

    )
    log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
    except ClientError as err:
        log.error("Failed to replace instance profile.")
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidAssociationID.NotFound":
            log.error(
                f"Association ID '{profile_association_id}' does not exist."
                "Please check the association ID and try again."
            )
        if error_code == "InvalidInstanceId":
            log.error(
                f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
                f"Please verify the instance ID and try again."
            )
        log.error(f"Full error:\n\t{err}")

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```

pub async fn reboot(&self, ec2: &EC2) -> Result<(), EC2Error> {
    if self.instance.is_some() {
        ec2.reboot_instance(self.instance_id()).await?;
        ec2.wait_for_instance_stopped(self.instance_id(), None)
            .await?;
        ec2.wait_for_instance_ready(self.instance_id(), None)
            .await?;
    }
}

```



```
    }
    Ok(())
}
```

```
pub async fn reboot_instance(&self, instance_id: &str) -> Result<(),
EC2Error> {
    tracing::info!("Rebooting instance {instance_id}");

    self.client
        .reboot_instances()
        .instance_ids(instance_id)
        .send()
        .await?;

    Ok(())
}
```

使用等待器 API 等待实例进入已停止和就绪状态。使用等待器 API 要求在 rust 文件中添加“use aws_sdk_ec2::client::Waiter”。

```
/// Wait for an instance to be ready and status ok (default wait 60 seconds)
pub async fn wait_for_instance_ready(
    &self,
    instance_id: &str,
    duration: Option<Duration>,
) -> Result<(), EC2Error> {
    self.client
        .wait_until_instance_status_ok()
        .instance_ids(instance_id)
        .wait(duration.unwrap_or(Duration::from_secs(60)))
        .await
        .map_err(|err| match err {
            WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                "Exceeded max time ({}s) waiting for instance to start.",
                exceeded.max_wait().as_secs()
            )),
            _ => EC2Error::from(err),
        })?;
    Ok(())
}
```

```
pub async fn wait_for_instance_stopped(
    &self,
    instance_id: &str,
    duration: Option<Duration>,
) -> Result<(), EC2Error> {
    self.client
        .wait_until_instance_stopped()
        .instance_ids(instance_id)
        .wait(duration.unwrap_or(Duration::from_secs(60)))
        .await
        .map_err(|err| match err {
            WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                "Exceeded max time ({}s) waiting for instance to stop.",
                exceeded.max_wait().as_secs(),
            )),
            _ => EC2Error::from(err),
        })?;
    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用[RebootInstances](#)于 Rust 的AWS SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，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 SDK 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。

将 **RegisterImage** 与 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 AMI。

```
aws ec2 register-image \  
  --name my-image \  
  --image-location amzn-s3-demo-bucket/myimage/image.manifest.xml
```

输出：

```
{  
  "ImageId": "ami-1234567890EXAMPLE"  
}
```

有关更多信息，请参阅《[亚马逊 EC2 用户指南](#)》中的[亚马逊系统映像 \(AMI\)](#)。

示例 2：使用根设备的快照注册 AMI

以下 register-image 示例使用 EBS 根卷的指定快照作为设备 /dev/xvda 注册 AMI。块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《[亚马逊 EC2 用户指南](#)》中的[亚马逊系统映像 \(AMI\)](#)。

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例在 Amazon S3 中使用指定的清单文件注册 AMI。

```
Register-EC2Image -ImageLocation amzn-s3-demo-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 \(V 4\) RegisterImage](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例在 Amazon S3 中使用指定的清单文件注册 AMI。

```
Register-EC2Image -ImageLocation amzn-s3-demo-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 \(V 5\) 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 SDK 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。

将 **RejectVpcPeeringConnection** 与 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

适用于 PowerShell V4 的工具

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 (V 4) [RejectVpcPeeringConnection](#)中的。

适用于 PowerShell V5 的工具

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 (V 5) [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 SDK 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。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)
- [创建具有私有子网和 NAT 网关的 VPC](#)
- [开始使用 Amazon VPC](#)

.NET

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。

```
/// <summary>
/// Release an Elastic IP address. After the Elastic IP address is released,
/// it can no longer be used.
/// </summary>
/// <param name="allocationId">The allocation Id of the Elastic IP address.</
param>
/// <returns>True if successful.</returns>
public async Task<bool> ReleaseAddress(string allocationId)
{
    try
    {
        var request = new ReleaseAddressRequest { AllocationId =
allocationId };

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

        return false;
    }
    catch (Exception ex)
    {
        _logger.LogError(
```

```

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

```

- 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](#) 中查找完整示例，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](#) 中查找完整示例，了解如何进行设置和运行。

```
//! Release an Elastic IP address.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::releaseAddress(const Aws::String &allocationID,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::EC2::EC2Client ec2(clientConfiguration);

    Aws::EC2::Model::ReleaseAddressRequest request;
    request.SetAllocationId(allocationID);

    Aws::EC2::Model::ReleaseAddressOutcome outcome = ec2.ReleaseAddress(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to release Elastic IP address " <<
            allocationID << ":" << outcome.GetError().GetMessage() <<
            std::endl;
    } else {
        std::cout << "Successfully released Elastic IP address " <<
            allocationID << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 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-Classical 发布弹性 IP 地址

此示例释放弹性 IP 地址以用于 EC2-Classical 中的实例。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](#) 中查找完整示例，了解如何进行设置和运行。

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

```

        ReleaseAddressRequest request = ReleaseAddressRequest.builder()
            .allocationId(allocId)
            .build();

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

        return response;
    }

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```

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

/**
 * Release an Elastic IP address.
 * @param {{ allocationId: string }} options
 */
export const main = async ({ allocationId }) => {
    const client = new EC2Client({});
    const command = new ReleaseAddressCommand({
        // You can also use PublicIp, but that is for EC2 classic which is being
        retired.
        AllocationId: allocationId,
    });

```

```
try {
    await client.send(command);
    console.log("Successfully released address.");
} catch (caught) {
    if (
        caught instanceof Error &&
        caught.name === "InvalidAllocationID.NotFound"
    ) {
        console.warn(`${caught.message}. Please provide a valid AllocationID.`);
    } else {
        throw caught;
    }
}
};
```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```
suspend fun releaseEC2AddressSc(allocId: String?) {
    val request =
        ReleaseAddressRequest {
            allocationId = allocId
        }

    Ec2Client.fromEnvironment { 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 Kotlin 的 AWS SDK API 参考。

PowerShell

适用于 PowerShell V4 的工具

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 (V 4) [ReleaseAddress](#)中的。

适用于 PowerShell V5 的工具

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 (V 5) [ReleaseAddress](#)中的。

Python

适用于 Python 的 SDK (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。

```

class ElasticIpWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) Elastic IP address
    actions using the client interface."""

    class ElasticIp:
        """Represents an Elastic IP and its associated instance."""

        def __init__(
            self, allocation_id: str, public_ip: str, instance_id: Optional[str]
            = None
        ) -> None:
            """
            Initializes the ElasticIp object.

            :param allocation_id: The allocation ID of the Elastic IP.
            :param public_ip: The public IP address of the Elastic IP.
            :param instance_id: The ID of the associated EC2 instance, if any.
            """
            self.allocation_id = allocation_id
            self.public_ip = public_ip
            self.instance_id = instance_id

        def __init__(self, ec2_client: Any) -> None:
            """
            Initializes the ElasticIpWrapper with an EC2 client.

            :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
            level
                               access to AWS EC2 services.
            """
            self.ec2_client = ec2_client
            self.elastic_ips: List[ElasticIpWrapper.ElasticIp] = []

        @classmethod
        def from_client(cls) -> "ElasticIpWrapper":
            """
            Creates an ElasticIpWrapper instance with a default EC2 client.

            :return: An instance of ElasticIpWrapper initialized with the default EC2
            client.
            """
            ec2_client = boto3.client("ec2")
            return cls(ec2_client)

```



```

def release(self, allocation_id: str) -> None:
    """
    Releases an Elastic IP address. After the Elastic IP address is released,
    it can no longer be used.

    :param allocation_id: The allocation ID of the Elastic IP to release.
    :raises ClientError: If the release fails, such as when the Elastic IP
    address is not found.
    """
    elastic_ip = self.get_elastic_ip_by_allocation(self.elastic_ips,
    allocation_id)
    if elastic_ip is None:
        logger.info(f"No Elastic IP found with allocation ID
    {allocation_id}.")
        return

    try:
        self.ec2_client.release_address(AllocationId=allocation_id)
        self.elastic_ips.remove(elastic_ip) # Remove the Elastic IP from the
    list
    except ClientError as err:
        if err.response["Error"]["Code"] == "InvalidAddress.NotFound":
            logger.error(
                f"Failed to release Elastic IP address {allocation_id} "
                "because it could not be found. Verify the Elastic IP address
            "
                "and ensure it is allocated to your account in the correct
            region "
                "before attempting to release it."
            )
            raise

```

- 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](#) 中查找完整示例，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)
  true
rescue StandardError => e
  puts("Error releasing Elastic IP address: #{e.message}")
  false
end
```

- 有关 API 的详细信息，请参阅 AWS SDK for Ruby API 参考 [ReleaseAddress](#) 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](#) 中查找完整示例，了解如何进行设置和运行。

```
pub async fn deallocate_ip_address(&self, allocation_id: &str) -> Result<(),
EC2Error> {
    self.client
        .release_address()
        .allocation_id(allocation_id)
        .send()
        .await?;
    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用 [ReleaseAddress](#) 于 Rust 的 AWS SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，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。

Swift

适用于 Swift 的 SDK

Note

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

```
import AWSEC2

/// Release an allocated Elastic IP.
///
/// - Parameter allocationId: The allocation ID of the Elastic IP to
///   release.
func releaseAddress(allocationId: String?) async {
    do {
        _ = try await ec2Client.releaseAddress(
            input: ReleaseAddressInput(
                allocationId: allocationId
            )
        )
    } catch {
        print("**** Unable to release the IP address:
\\(error.localizedDescription)")
    }
}
```

- 有关 API 的详细信息，请参阅适用于 S wift 的AWS SDK API 参考[ReleaseAddress](#)中。

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 SDK 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。

将 **ReleaseHosts** 与 CLI 配合使用

以下代码示例演示如何使用 ReleaseHosts。

CLI

AWS CLI

从账户中释放专属主机

从您的账户中释放专属主机。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

适用于 PowerShell V4 的工具

示例 1：此示例释放给定的主机 ID 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 (V 4) [ReleaseHosts](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例释放给定的主机 ID 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 (V 5) [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 SDK 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

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。

```
/// <summary>
/// Replace the profile associated with a running instance. After the profile
is replaced, the instance
/// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
/// used to restart the Python web server.
/// </summary>
/// <param name="instanceId">The Id of the instance to update.</param>
/// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
/// <param name="associationId">The Id of the existing profile association
for the instance.</param>
/// <returns>Async task.</returns>
public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
{
    try
    {
        await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
            new ReplaceIamInstanceProfileAssociationRequest()
            {
                AssociationId = associationId,
                IamInstanceProfile = new IamInstanceProfileSpecification()
                {
                    Name = credsProfileName
                }
            });
        // Allow time before resetting.
        Thread.Sleep(25000);
    }
}
```

```

        await _amazonEc2.RebootInstancesAsync(
            new RebootInstancesRequest(new List<string>() { instanceId }));
        Thread.Sleep(25000);
        var instanceReady = false;
        var retries = 5;
        while (retries-- > 0 && !instanceReady)
        {
            var instancesPaginator =
                _amazonSsm.Paginators.DescribeInstanceInformation(
                    new DescribeInstanceInformationRequest());
            // Get the entire list using the paginator.
            await foreach (var instance in
instancesPaginator.InstanceInformationList)
            {
                instanceReady = instance.InstanceId == instanceId;
                if (instanceReady)
                {
                    break;
                }
            }
        }
        Console.WriteLine("Waiting for instance to be running.");
        await WaitForInstanceState(instanceId, InstanceStateName.Running);
        Console.WriteLine("Instance ready.");
        Console.WriteLine($"Sending restart command to instance
{instanceId}");
        await _amazonSsm.SendCommandAsync(
            new SendCommandRequest()
            {
                InstanceIds = new List<string>() { instanceId },
                DocumentName = "AWS-RunShellScript",
                Parameters = new Dictionary<string, List<string>>()
                {
                    {
                        "commands",
                        new List<string>() { "cd / && sudo python3 server.py
80" }
                    }
                }
            });
        Console.WriteLine($"Restarted the web server on instance
{instanceId}");
    }
    catch (AmazonEC2Exception ec2Exception)

```



```

        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

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

```

- 有关 API 的详细信息，请参阅 AWS SDK for .NET API 参考 [ReplacelamInstanceProfileAssociation](#) 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[ReplacelamInstanceProfileAssociation](#)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](#) 中查找完整示例，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[ReplacelamInstanceProfileAssociation](#)中的。

Python

适用于 Python 的 SDK (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。

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 AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
```

```

self.ec2_client = ec2_client
self.ssm_client = ssm_client
self.iam_client = iam_client
sts_client = boto3.client("sts")
self.account_id = sts_client.get_caller_identity()["Account"]

self.key_pair_name = f"{resource_prefix}-key-pair"
self.launch_template_name = f"{resource_prefix}-template-"
self.group_name = f"{resource_prefix}-group"

# Happy path
self.instance_policy_name = f"{resource_prefix}-pol"
self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"

# Failure mode
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

def replace_instance_profile(
    self,
    instance_id: str,
    new_instance_profile_name: str,
    profile_association_id: str,
) -> None:
    """
    Replaces the profile associated with a running instance. After the
    profile is
        replaced, the instance is rebooted to ensure that it uses the new
    profile. When
        the instance is ready, Systems Manager is used to restart the Python web
    server.

    :param instance_id: The ID of the instance to restart.
    :param new_instance_profile_name: The name of the new profile to
    associate with
        the specified instance.
    :param profile_association_id: The ID of the existing profile association
    for the
        instance.
    """
    try:

```

```

self.ec2_client.replace_iam_instance_profile_association(
    IamInstanceProfile={"Name": new_instance_profile_name},
    AssociationId=profile_association_id,
)
log.info(
    "Replaced instance profile for association %s with profile %s.",
    profile_association_id,
    new_instance_profile_name,
)
time.sleep(5)

self.ec2_client.reboot_instances(InstanceIds=[instance_id])
log.info("Rebooting instance %s.", instance_id)
waiter = self.ec2_client.get_waiter("instance_running")
log.info("Waiting for instance %s to be running.", instance_id)
waiter.wait(InstanceIds=[instance_id])
log.info("Instance %s is now running.", instance_id)

self.ssm_client.send_command(
    InstanceIds=[instance_id],
    DocumentName="AWS-RunShellScript",
    Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
)
log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
except ClientError as err:
    log.error("Failed to replace instance profile.")
    error_code = err.response["Error"]["Code"]
    if error_code == "InvalidAssociationID.NotFound":
        log.error(
            f"Association ID '{profile_association_id}' does not exist."
            "Please check the association ID and try again."
        )
    if error_code == "InvalidInstanceId":
        log.error(
            f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
            f"Please verify the instance ID and try again."
        )
    log.error(f"Full error:\n\t{err}")

```

- 有关 API 的详细信息，请参阅适用[ReplaceNetworkAclAssociation](#)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 SDK 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。

将 **ReplaceNetworkAclAssociation** 与 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 aiclassoc-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

适用于 PowerShell V4 的工具

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：

```
aaclassoc-87654321
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 4) [ReplaceNetworkAclAssociation](#) 中的。

适用于 PowerShell V5 的工具

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：

```
aaclassoc-87654321
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) [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 SDK 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。

将 **ReplaceNetworkAclEntry** 与 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

此示例将替换指定网络 ACL 的一个条目。新规则 100 允许来自 UDP 端口 53 (DNS) 上的 203.0.113.12/24 的入口流量进入任何关联的子网。

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

适用于 PowerShell V4 的工具

示例 1：此示例替换指定网络 ACL 的指定条目。新规则支持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 \(V 4\)](#) [ReplaceNetworkAclEntry](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例替换指定网络 ACL 的指定条目。新规则支持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 \(V 5\)](#) [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 SDK 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。

将 **ReplaceRoute** 与 CLI 配合使用

以下代码示例演示如何使用 `ReplaceRoute`。

CLI

AWS CLI

替换路由

此示例将替换指定路由表中的指定路由。新路由与指定的 CIDR 匹配，并将流量发送到指定的虚拟私有网关。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

适用于 PowerShell V4 的工具

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 (V 4) [ReplaceRoute](#) 中的。

适用于 PowerShell V5 的工具

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 (V 5) [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 SDK 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。

将 **ReplaceRouteTableAssociation** 与 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

适用于 PowerShell V4 的工具

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 (V 4) [ReplaceRouteTableAssociation](#)中的。

适用于 PowerShell V5 的工具

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 \(V 5\) `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 SDK 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。

将 `ReportInstanceStatus` 与 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

适用于 PowerShell V4 的工具

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 \(V 4\) `ReportInstanceStatus`](#) 中的。

适用于 PowerShell V5 的工具

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 \(V 5\) 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 SDK 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。

将 **RequestSpotFleet** 与 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竞价型实例集

此示例命令创建一个竞价型实例集请求，其中包含两个仅因子网而异的启动规范。竞价型实例集启动指定子网中价格最低的实例。如果在默认 VPC 中启动实例，则实例在默认情况下会收到一个公有 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，您无法在竞价型实例集请求中指定来自相同可用区的不同子网。

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"
}

```

请求可用区中价格最低的竞价型实例集

此示例命令创建一个竞价型实例集请求，其中包含两个仅因可用区而异的启动规范。竞价型实例集启动指定可用区中价格最低的实例。如果您的账户仅支持 EC2-VPC，则 Amazon EC2 将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，则亚马逊 EC2 会在可用区中启动 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

此示例命令将公有地址分配给在非默认 VPC 中启动的实例。请注意，指定网络接口时，您必须包括使用网络接口的子网 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"
    }
}
]
}

```

使用多样化分配策略请求竞价型实例集

此示例命令创建一个竞价型实例集请求，该请求使用多样化分配策略启动 30 个实例。启动规范因实例类型而异。竞价型实例集在启动规范间分配实例，以便每种类型有 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 Elastic Compute Cloud 用户指南》中的“竞价型实例集请求”。

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例在价格最低的可用区中为指定的实例类型创建竞价型实例集请求。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中启动 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 (V 4) [RequestSpotFleet](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例在价格最低的可用区中为指定的实例类型创建竞价型实例集请求。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中启动 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 $1c
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 5\) 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 SDK 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。

将 **RequestSpotInstances** 与 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，则亚马逊 EC2 将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
```

Specification.json :

```
{
  "ImageId": "ami-1a2b3c4d",
  "KeyName": "my-key-pair",
  "SecurityGroupIds": [ "sg-1a2b3c4d" ],
  "InstanceType": "m3.medium",
  "Placement": {
    "AvailabilityZone": "us-west-2a"
  },
}
```

```
"IamInstanceProfile": {
  "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
}
}
```

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 在指定子网中 EC2 启动实例。如果 VPC 是一个非默认 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
```

Specification.json：

```
{
  "ImageId": "ami-1a2b3c4d",
  "SecurityGroupIds": [ "sg-1a2b3c4d" ],
  "InstanceType": "m3.medium",
  "SubnetId": "subnet-1a2b3c4d",
  "IamInstanceProfile": {
    "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
  }
}
```

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，指定网络接口时，您必须包括使用网络接口的子网 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
```

Specification.json :

```

{
    "ImageId": "ami-1a2b3c4d",
    "KeyName": "my-key-pair",
    "InstanceType": "m3.medium",
    "NetworkInterfaces": [
        {
            "DeviceIndex": 0,
            "SubnetId": "subnet-1a2b3c4d",
            "Groups": [ "sg-1a2b3c4d" ],

```

```

        "AssociatePublicIpAddress": true
    }
],
    "IamInstanceProfile": {
        "Arn": "arn:aws:iam::123456789012:instance-profile/my-iam-role"
    }
}

```

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例在指定的子网中请求一次性竞价型实例。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 指定该安全组。指定网络接口时，您必须包括使用网络接口的子网 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 \(V 4\) RequestSpotInstances](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例在指定的子网中请求一次性竞价型实例。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 指定该安全组。指定网络接口时，您必须包括使用网络接口的子网 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 \(V 5\) 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 SDK 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。

将 **ResetImageAttribute** 与 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，AMIs 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

适用于 PowerShell V4 的工具

示例 1：此示例将“`launchPermission`”属性重置为其默认值。默认情况下，AMIs 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 \(V 4\) ResetImageAttribute](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例将“`launchPermission`”属性重置为其默认值。默认情况下，AMIs 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 \(V 5\) 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 SDK 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。

将 **ResetInstanceAttribute** 与 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

此示例重置指定实例的 `sourceDestCheck` 属性。该实例必须在 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
```

重置虚拟磁盘属性

此示例重置指定实例的 `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

适用于 PowerShell V4 的工具

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“ebsOptimized”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 (V 4) [ResetInstanceAttribute](#)中的。

适用于 PowerShell V5 的工具

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“ebsOptimized”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 \(V 5\)](#) [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 SDK 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。

将 **ResetNetworkInterfaceAttribute** 与 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 source/destination 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

适用于 PowerShell V4 的工具

示例 1：此示例重置对指定网络接口的 source/destination 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 (V 4) [ResetNetworkInterfaceAttribute](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例重置对指定网络接口的 source/destination 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 (V 5) [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 SDK 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。

将 **ResetSnapshotAttribute** 与 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

适用于 PowerShell V4 的工具

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 (V 4) [ResetSnapshotAttribute](#)中的。

适用于 PowerShell V5 的工具

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 (V 5) [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 SDK 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。

将 **RevokeSecurityGroupEgress** 与 CLI 配合使用

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

CLI

AWS CLI

示例 1：删除允许指向特定地址范围的出站流量的规则

以下 `revoke-security-group-egress` 示例命令删除授予对 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：删除允许指向特定安全组的出站流量的规则

以下 `revoke-security-group-egress` 示例命令删除授予对 TCP 端口 80 上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

适用于 PowerShell V4 的工具

示例 1：此示例删除了指定安全组的 EC2-VPC 规则。这将撤销对 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：此示例撤销对 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 \(V 4\)](#) [RevokeSecurityGroupEgress](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例删除了指定安全组的 EC2-VPC 规则。这将撤销对 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：此示例撤销对 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 \(V 5\)](#) [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 SDK 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。

将 **RevokeSecurityGroupIngress** 与 CLI 配合使用

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

CLI

AWS CLI

示例 1：从安全组中删除规则

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，此命令不会产生任何输出。

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 Destination Unreachable: Fragmentation Needed and Don't Fragment was Set (类型 3，代码 4) 的入站规则。

```
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，此命令不会产生任何输出。

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

适用于 PowerShell V4 的工具

示例 1：此示例撤消了 EC2-VPC 的指定安全组的指定地址范围内对 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：此示例撤消了 EC2-Classic 的指定安全组的指定地址范围内对 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 (V 4) [RevokeSecurityGroupIngress](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例撤消了 EC2-VPC 的指定安全组的指定地址范围内对 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：此示例撤消了 EC2-Classic 的指定安全组的指定地址范围内对 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 \(V 5\) `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 SDK 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。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)
- [开始使用 Amazon VPC](#)

.NET

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。

```
/// <summary>
/// Create and run an EC2 instance.
/// </summary>
/// <param name="ImageId">The image Id of the image used as a basis for the
/// EC2 instance.</param>
```

```

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

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

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

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

```

```
        throw;
    }
}
```

- 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](#) 中查找完整示例，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 getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
```

```

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

# Retrieve the calling parameters.
while getopts "i:t:k:s:c:h" option; do
    case "${option}" in
        i) image_id="${OPTARG}" ;;
        t) instance_type="${OPTARG}" ;;
        k) key_pair_name="${OPTARG}" ;;
        s) security_group_id="${OPTARG}" ;;
        c) count="${OPTARG}" ;;
        h)
            usage
            return 0
            ;;
        \?)
            echo "Invalid parameter"
            usage
            return 1
            ;;
    esac
done
export OPTIND=1

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

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

```

```

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](#) 中查找完整示例，了解如何进行设置和运行。

```

//! Launch an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
    \param instanceName: A name for the EC2 instance.
    \param amiId: An Amazon Machine Image (AMI) identifier.
    \param[out] instanceID: String to return the instance ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::runInstance(const Aws::String &instanceName,
                              const Aws::String &amiId,
                              Aws::String &instanceID,
                              const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::RunInstancesRequest runRequest;
    runRequest.SetImageId(amiId);
    runRequest.SetInstanceType(Aws::EC2::Model::InstanceType::t1_micro);
    runRequest.SetMinCount(1);
    runRequest.SetMaxCount(1);

    Aws::EC2::Model::RunInstancesOutcome runOutcome = ec2Client.RunInstances(
        runRequest);
    if (!runOutcome.IsSuccess()) {
        std::cerr << "Failed to launch EC2 instance " << instanceName <<
            " based on ami " << amiId << ":" <<
            runOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    const Aws::Vector<Aws::EC2::Model::Instance> &instances =
runOutcome.GetResult().GetInstances();

```



```

    if (instances.empty()) {
        std::cerr << "Failed to launch EC2 instance " << instanceName <<
            " based on ami " << amiId << ":" <<
            runOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    instanceID = instances[0].GetInstanceId();

    return true;
}

```

- 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",
        "PrivateAddresses": [
            {
                "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。

有关块储存设备映射的更多信息，请参阅 Amazon EC2 用户指南中的[块储存设备映射](#)。

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 用户指南中的使用实例 EC2 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。

有关突发性能实例的更多信息，请参阅 Amazon EC2 用户指南中的[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](#) 中查找完整示例，了解如何进行设置和运行。

```
/**
```

```

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

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

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

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

/**
 * Create new EC2 instances.
 * @param {{
 *   keyName: string,
 *   securityGroupIds: string[],
 *   imageId: string,
 *   instanceType: import('@aws-sdk/client-ec2')._InstanceType,
 *   minCount?: number,
 *   maxCount?: number }} options
 */
export const main = async ({
  keyName,
  securityGroupIds,
  imageId,
  instanceType,
  minCount = "1",
  maxCount = "1",
}) => {
  const client = new EC2Client({});
  minCount = Number.parseInt(minCount);
  maxCount = Number.parseInt(maxCount);
  const command = new RunInstancesCommand({
    // Your key pair name.
    KeyName: keyName,
    // Your security group.
    SecurityGroupIds: securityGroupIds,
    // An Amazon Machine Image (AMI). There are multiple ways to search for AMIs.
    // For more information, see:
    // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/finding-an-ami.html
    ImageId: imageId,
```



```

    // An instance type describing the resources provided to your instance. There
    // are multiple
    // ways to search for instance types. For more information see:
    // https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-
    // discovery.html
    InstanceType: instanceType,
    // Availability Zones have capacity limitations that may impact your ability
    // to launch instances.
    // The `RunInstances` operation will only succeed if it can allocate at least
    // the `MinCount` of instances.
    // However, EC2 will attempt to launch up to the `MaxCount` of instances,
    // even if the full request cannot be satisfied.
    // If you need a specific number of instances, use `MinCount` and `MaxCount`
    // set to the same value.
    // If you want to launch up to a certain number of instances, use `MaxCount`
    // and let EC2 provision as many as possible.
    // If you require a minimum number of instances, but do not want to exceed a
    // maximum, use both `MinCount` and `MaxCount`.
    MinCount: minCount,
    MaxCount: maxCount,
  });

  try {
    const { Instances } = await client.send(command);
    const instanceList = Instances.map(
      (instance) => `• ${instance.InstanceId}`,
    ).join("\n");
    console.log(`Launched instances:\n${instanceList}`);
  } catch (caught) {
    if (caught instanceof Error && caught.name === "ResourceCountExceeded") {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};

```

- 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](#) 中查找完整示例，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.fromEnvironment { 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 Kotlin 的 AWS SDK API 参考。

PowerShell

适用于 PowerShell V4 的工具

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，请定义块设备映射并将其添加到命令中。此示例添加实例存储卷。

```
$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 Windows 中的一个 AMIs，请使用获取其 AMI ID `Get-EC2ImageByName`。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：将实例启动到指定的专属主机环境中。

```
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，并将键为 `webserver` 和值为 `production` 的标签应用于实例。该请求还向创建的卷（此示例中为每个实例的根卷）应用键为 `cost-center` 且值为 `cc123` 的标签。

```
$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 (V 4) [RunInstances](#) 中的。

适用于 PowerShell V5 的工具

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，请定义块设备映射并将其添加到命令中。此示例添加实例存储卷。

```
$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 Windows 中的一个 AMIs，请使用 Get-SSMLatestEC2Image 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-SSMLatestEC2Image -Path 'ami-windows-latest' -ImageName
'WINDOWS*2016*English*Core*BASE').Value

New-EC2Instance -ImageId $ami ...
```

示例 5：将实例启动到指定的专属主机环境中。

```
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，并将键为 webserver 和值为 production 的标签应用于实例。该请求还向创建的卷（此示例中为每个实例的根卷）应用键为 cost-center 且值为 cc123 的标签。

```
$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
```

示例 7：此示例验证使用 DryRun 参数启动 EC2 实例的权限，而无需实际创建实例。注：这在操作成功时会引发异常，这是预期行为。

```
New-EC2Instance -ImageId 'ami-12345678' -InstanceType 't2.micro' -KeyName 'my-
key-pair' -Region 'us-west-2' -DryRun $true
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) [RunInstances](#) 中的。

Python

适用于 Python 的 SDK (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。

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
        objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
        EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```

def create(
    self,
    image_id: str,
    instance_type: str,
    key_pair_name: str,
    security_group_ids: Optional[List[str]] = None,
) -> List[Dict[str, Any]]:
    """
    Creates a new EC2 instance in the default VPC of the current account.

    The instance starts immediately after it is created.

    :param image_id: The ID of the Amazon Machine Image (AMI) to use for the
    instance.
    :param instance_type: The type of instance to create, such as 't2.micro'.
    :param key_pair_name: The name of the key pair to use for SSH access.
    :param security_group_ids: A list of security group IDs to associate with
    the instance.

    If not specified, the default security group
    of the VPC is used.

    :return: A list of dictionaries representing Boto3 Instance objects
    representing the newly created instances.
    """
    try:
        instance_params = {
            "ImageId": image_id,
            "InstanceType": instance_type,
            "KeyName": key_pair_name,
        }
        if security_group_ids is not None:
            instance_params["SecurityGroupIds"] = security_group_ids

        response = self.ec2_client.run_instances(
            **instance_params, MinCount=1, MaxCount=1
        )
        instance = response["Instances"][0]
        self.instances.append(instance)
        waiter = self.ec2_client.get_waiter("instance_running")
        waiter.wait(InstanceIds=[instance["InstanceId"]])
    except ClientError as err:
        params_str = "\n\t".join(
            f"{key}: {value}" for key, value in instance_params.items()
        )

```

```

        logger.error(
            f"Failed to complete instance creation request.\nRequest details:
{params_str}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "InstanceLimitExceeded":
            logger.error(
                (
                    f"Insufficient capacity for instance type
'{instance_type}'. "
                    "Terminate unused instances or contact AWS Support for a
limit increase."
                )
            )
        if error_code == "InsufficientInstanceCapacity":
            logger.error(
                (
                    f"Insufficient capacity for instance type
'{instance_type}'. "
                    "Select a different instance type or launch in a
different availability zone."
                )
            )
        raise
    return self.instances

```

- 有关 API 的详细信息，请参阅适用[RunInstances](#)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](#) 中查找完整示例，了解如何进行设置和运行。

```

pub async fn create_instance<'a>(
    &self,

```



```

        image_id: &'a str,
        instance_type: InstanceType,
        key_pair: &'a KeyPairInfo,
        security_groups: Vec<&'a SecurityGroup>,
    ) -> Result<String, EC2Error> {
        let run_instances = self
            .client
            .run_instances()
            .image_id(image_id)
            .instance_type(instance_type)
            .key_name(
                key_pair
                    .key_name()
                    .ok_or_else(|| EC2Error::new("Missing key name when launching
instance"))?),
            )
            .set_security_group_ids(Some(
                security_groups
                    .iter()
                    .filter_map(|sg| sg.group_id.clone())
                    .collect(),
            ))
            .min_count(1)
            .max_count(1)
            .send()
            .await?;

        if run_instances.instances().is_empty() {
            return Err(EC2Error::new("Failed to create instance"));
        }

        let instance_id = run_instances.instances()[0].instance_id().unwrap();
        let response = self
            .client
            .create_tags()
            .resources(instance_id)
            .tags(
                Tag::builder()
                    .key("Name")
                    .value("From SDK Examples")
                    .build(),
            )
            .send()
            .await;

```

```

match response {
    Ok(_) => tracing::info!("Created {instance_id} and applied tags."),
    Err(err) => {
        tracing::info!("Error applying tags to {instance_id}: {err:?}");
        return Err(err.into());
    }
}

tracing::info!("Instance is created.");

Ok(instance_id.to_string())
}

```

- 有关 API 的详细信息，请参阅适用[RunInstances](#)于 Rust 的 AWS SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

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

```

" Create tags for resource created during instance launch. "
DATA lt_tagsspecifications TYPE /aws1/
cl_ec2tagsspecification=>tt_tagsspecificationlist.
DATA ls_tagsspecifications LIKE LINE OF lt_tagsspecifications.
ls_tagsspecifications = NEW /aws1/cl_ec2tagsspecification(
    iv_resourcetype = 'instance'
    it_tags = VALUE /aws1/cl_ec2tag=>tt_taglist(
        ( NEW /aws1/cl_ec2tag( iv_key = 'Name' iv_value = iv_tag_value ) )
    ) ).
APPEND ls_tagsspecifications TO lt_tagsspecifications.

TRY.
    " Create/launch Amazon Elastic Compute Cloud (Amazon EC2) instance. "

```

```

        oo_result = lo_ec2->runinstances(
is returned for testing purposes. "
        iv_imageid = iv_ami_id
        iv_instancetype = 't3.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。

Swift

适用于 Swift 的 SDK

Note

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

```

import AWSEC2

/// Create and return a new EC2 instance.
///
/// - Parameters:
///   - imageId: The image ID of the AMI to use when creating the instance.
///   - instanceType: The type of instance to create.
///   - keyPairName: The RSA key pair's name to use to secure the instance.
///   - securityGroups: The security group or groups to add the instance
///     to.
///
/// - Returns: The EC2 instance as an `EC2ClientTypes.Instance` object.
func runInstance(imageId: String, instanceType: EC2ClientTypes.InstanceType,

```

```

        keyPairName: String, securityGroups: [String]?) async ->
EC2ClientTypes.Instance? {
    do {
        let output = try await ec2Client.runInstances(
            input: RunInstancesInput(
                imageId: imageId,
                instanceType: instanceType,
                keyName: keyPairName,
                maxCount: 1,
                minCount: 1,
                securityGroupIds: securityGroups
            )
        )

        guard let instances = output.instances else {
            print("*** Unable to create the instance.")
            return nil
        }

        return instances[0]
    } catch {
        print("*** Error creating the instance:
\\(error.localizedDescription)")
        return nil
    }
}

```

- 有关 API 的详细信息，请参阅适用于 Swift 的 AWS SDK API 参考[RunInstances](#)中。

有关 Swift AWS SDK 开发者指南和代码示例的完整列表，请参阅[使用 AWS SDK 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。

将 **RunScheduledInstances** 与 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
```

Launch-specification.json :

```
{
  "ImageId": "ami-12345678",
  "KeyName": "my-key-pair",
  "InstanceType": "c4.large",
  "NetworkInterfaces": [
    {
      "DeviceIndex": 0,
      "SubnetId": "subnet-12345678",
      "AssociatePublicIpAddress": true,
      "Groups": ["sg-12345678"]
    }
  ],
  "IamInstanceProfile": {
    "Name": "my-iam-role"
  }
}
```

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
```

Launch-specification.json :

```
{
  "ImageId": "ami-12345678",
  "KeyName": "my-key-pair",
  "SecurityGroupIds": ["sg-12345678"],
  "InstanceType": "c4.large",
  "Placement": {
    "AvailabilityZone": "us-west-2b"
  }
  "IamInstanceProfile": {
    "Name": "my-iam-role"
  }
}
```

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

适用于 PowerShell V4 的工具

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 (V 4) [RunScheduledInstances](#)中的。

适用于 PowerShell V5 的工具

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 \(V 5\) `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 SDK 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：

- [了解基本功能](#)

.NET

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。

```
/// <summary>  
/// Start an EC2 instance.  
/// </summary>
```

```

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

            await _amazonEC2.StartInstancesAsync(request);

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

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

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
        InstanceStateName stateName)

```



```

{
    var request = new DescribeInstancesRequest
    {
        InstanceIds = new List<string> { instanceId }
    };

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

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

    return hasState;
}

```

- 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](#) 中查找完整示例，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 getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function ec2_start_instances"
        echo "Starts one or more Amazon Elastic Compute Cloud (Amazon EC2)
instances."
        echo "  -i instance_id - The ID(s) of the instance(s) to start (comma-
separated)."
        echo "  -h - Display help."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "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."
    fi
}

```

```

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](#) 中查找完整示例，了解如何进行设置和运行。

```

//! Start an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
    \param instanceID: An EC2 instance ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
 */
bool AwsDoc::EC2::startInstance(const Aws::String &instanceId,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::StartInstancesRequest startRequest;

```

```

    startRequest.AddInstanceIds(instanceId);
    startRequest.SetDryRun(true);

    Aws::EC2::Model::StartInstancesOutcome dryRunOutcome =
    ec2Client.StartInstances(startRequest);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to start instance. A dry run should trigger an
error."
            << std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
        Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to start instance " << instanceId << ": "
            << dryRunOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    startRequest.SetDryRun(false);
    Aws::EC2::Model::StartInstancesOutcome startInstancesOutcome =
    ec2Client.StartInstances(startRequest);

    if (!startInstancesOutcome.IsSuccess()) {
        std::cout << "Failed to start instance " << instanceId << ": " <<
            startInstancesOutcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully started instance " << instanceId <<
            std::endl;
    }

    return startInstancesOutcome.IsSuccess();
}

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```
/**
 * Starts an Amazon EC2 instance asynchronously and waits until it is in the
 * "running" state.
 */
```

```

    * @param instanceId the ID of the instance to start
    * @return a {@link CompletableFuture} that completes when the instance has
    *         been started and is in the "running" state, or exceptionally if an error occurs
    */
    public CompletableFuture<Void> startInstanceAsync(String instanceId) {
        StartInstancesRequest startRequest = StartInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
            .client(getAsyncClient())
            .build();

        DescribeInstancesRequest describeRequest =
        DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

        logger.info("Starting instance " + instanceId + " and waiting for it to
run.");
        CompletableFuture<Void> resultFuture = new CompletableFuture<>();
        return getAsyncClient().startInstances(startRequest)
            .thenCompose(response ->
                ec2Waiter.waitUntilInstanceRunning(describeRequest)
            )
            .thenAccept(waiterResponse -> {
                logger.info("Successfully started instance " + instanceId);
                resultFuture.complete(null);
            })
            .exceptionally(throwable -> {
                resultFuture.completeExceptionally(new RuntimeException("Failed
to start instance: " + throwable.getMessage(), throwable));
                return null;
            });
    }
}

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```
import { EC2Client, StartInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Starts an Amazon EBS-backed instance that you've previously stopped.
 * @param {{ instanceIds }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new StartInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { StartingInstances } = await client.send(command);
    const instanceIdList = StartingInstances.map(
      (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Starting instances:");
    console.log(instanceIdList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```


- 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](#) 中查找完整示例，了解如何进行设置和运行。

```
suspend fun startInstanceSc(instanceId: String) {
    val request =
        StartInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.startInstances(request)
        println("Waiting until instance $instanceId starts. This will take a few
minutes.")
        ec2.waitForInstanceRunning {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully started instance $instanceId")
    }
}
```

- 有关 API 的详细信息，请参阅适用[StartInstances](#)于 Kotlin 的 AWS SDK API 参考。

PowerShell

适用于 PowerShell V4 的工具

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 \(V 4\) StartInstances](#) 中的。

适用于 PowerShell V5 的工具

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
```

示例 5：此示例验证了在不实际启动 EC2 实例的情况下使用 DryRun 参数启动实例的权限。
注：这在操作成功时会引发异常，这是预期行为。

```
Start-EC2Instance -InstanceId 'i-0abcdef123456' -Region 'us-west-1' -DryRun $true
```

- 有关 API 的详细信息，请参阅 AWS Tools for PowerShell Cmdlet 参考 (V 5) [StartInstances](#) 中的。

Python

适用于 Python 的 SDK (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。

```
class EC2InstanceWrapper:  
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions  
    using the client interface."""
```

```

def __init__(
    self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
) -> None:
    """
    Initializes the EC2InstanceWrapper with an EC2 client and optional
    instances.

    :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
level
                        access to AWS EC2 services.
    :param instances: A list of dictionaries representing Boto3 Instance
objects. These are high-level objects that
                        wrap instance actions.
    """
    self.ec2_client = ec2_client
    self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def start(self) -> Optional[Dict[str, Any]]:
        """
        Starts instances and waits for them to be in a running state.

        :return: The response to the start request.
        """
        if not self.instances:
            logger.info("No instances to start.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            start_response =
self.ec2_client.start_instances(InstanceIds=instance_ids)

```

```

        waiter = self.ec2_client.get_waiter("instance_running")
        waiter.wait(InstanceIds=instance_ids)
        return start_response
    except ClientError as err:
        logger.error(
            f"Failed to start instance(s): {' '.join(map(str,
instance_ids))}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "IncorrectInstanceState":
            logger.error(
                "Couldn't start instance(s) because they are in an incorrect
state. "
                "Ensure the instances are in a stopped state before starting
them."
            )
            raise

```

- 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](#) 中查找完整示例，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.'
  true
rescue StandardError => e
  puts "Error starting instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.

```

```

    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
        'i-123abc us-west-2'
    exit 1
# If no values are specified at the command prompt, use these default values.
# Replace us-west-2 with the AWS Region you're using for Amazon EC2.
elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
# Otherwise, use the values as specified at the command prompt.
else
    instance_id = ARGV[0]
    region = ARGV[1]
end

ec2_client = Aws::EC2::Client.new(region: region)

puts "Attempting to start instance '#{instance_id}' " \
    '(this might take a few minutes)... '
return if instance_started?(ec2_client, instance_id)

puts 'Could not start instance.'
end

run_me if $PROGRAM_NAME == __FILE__

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

按 EC2 实例 ID 启动实例。

```

pub async fn start_instance(&self, instance_id: &str) -> Result<(), EC2Error>
{

```

```

        tracing::info!("Starting instance {instance_id}");

        self.client
            .start_instances()
            .instance_ids(instance_id)
            .send()
            .await?;

        tracing::info!("Started instance.");

        Ok(())
    }

```

使用等待器 API 等待实例进入就绪且状态正常的状态。使用等待器 API 要求在 rust 文件中添加“use aws_sdk_ec2::client::Waiter”。

```

/// Wait for an instance to be ready and status ok (default wait 60 seconds)
pub async fn wait_for_instance_ready(
    &self,
    instance_id: &str,
    duration: Option<Duration>,
) -> Result<(), EC2Error> {
    self.client
        .wait_until_instance_status_ok()
        .instance_ids(instance_id)
        .wait(duration.unwrap_or(Duration::from_secs(60)))
        .await
        .map_err(|err| match err {
            WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                "Exceeded max time ({}s) waiting for instance to start.",
                exceeded.max_wait().as_secs()
            )),
            _ => EC2Error::from(err),
        })?;
    Ok(())
}

```

- 有关 API 的详细信息，请参阅适用[StartInstances](#)于 Rust 的AWS SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

 Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，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。

Swift

适用于 Swift 的 SDK

Note

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

```

import AWSEC2

/// Start the specified instance.
///
/// - Parameters:
///   - instanceId: The ID of the instance to start.
///   - waitUntilStarted: If `true`, execution waits until the instance
///     has started. Otherwise, execution continues and the instance starts
///     asynchronously.
///
/// - Returns: `true` if the image is successfully started (or is left to
///   start asynchronously). `false` if the instance doesn't start.
func startInstance(instanceId: String, waitUntilStarted: Bool = false) async
-> Bool {
    let instanceList = [instanceId]

    do {
        _ = try await ec2Client.startInstances(
            input: StartInstancesInput(
                instanceIds: instanceList
            )
        )

        if waitUntilStarted {
            print("Waiting for the instance to start...")
        }
    }
}

```

```
        let waitOptions = WaiterOptions(maxWaitTime: 60.0)
        let output = try await ec2Client.waitUntilInstanceRunning(
            options: waitOptions,
            input: DescribeInstancesInput(
                instanceIds: instanceList
            )
        )
        switch output.result {
        case .success:
            return true
        case .failure:
            return false
        }
    } else {
        return true
    }
} catch {
    print("*** Unable to start the instance:
\\(error.localizedDescription)")
    return false
}
}
```

- 有关 API 的详细信息，请参阅适用于 Swift 的 AWS SDK API 参考[StartInstances](#)中。

有关 Swift AWS SDK 开发者指南和代码示例的完整列表，请参阅[使用 AWS SDK 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：

- [了解基本功能](#)

.NET

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。

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

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

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

        throw;
    }
}
```

```

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

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
        InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

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

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

        return hasState;
    }

```

- 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](#) 中查找完整示例，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 getopt command in a function.

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

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

```

        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](#) 中查找完整示例，了解如何进行设置和运行。

```

///
//! Stop an EC2 instance.
/*!
    \param instanceID: An EC2 instance ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::stopInstance(const Aws::String &instanceId,
                               const Aws::Client::ClientConfiguration
                               &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::StopInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::StopInstancesOutcome dryRunOutcome =
    ec2Client.StopInstances(request);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to stop instance. A dry run should trigger an
error."
            << std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
        Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to stop instance " << instanceId << ": "
            << dryRunOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::StopInstancesOutcome outcome =
    ec2Client.StopInstances(request);


```

```
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to stop instance " << instanceId << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully stopped instance " << instanceId <<
            std::endl;
    }

    return outcome.IsSuccess();
}

void PrintUsage() {
    std::cout << "Usage: run_start_stop_instance <instance_id> <start|stop>" <<
        std::endl;
}
```

- 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：让 Amazon 实例处于休眠状态 EC2

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](#) 中查找完整示例，了解如何进行设置和运行。

```
/**
 * Stops the EC2 instance with the specified ID asynchronously and waits for
 * the instance to stop.
 *
 * @param instanceId the ID of the EC2 instance to stop
 * @return a {@link CompletableFuture} that completes when the instance has
 * been stopped, or exceptionally if an error occurs
 */
public CompletableFuture<Void> stopInstanceAsync(String instanceId) {
    StopInstancesRequest stopRequest = StopInstancesRequest.builder()
        .instanceIds(instanceId)
        .build();

    DescribeInstancesRequest describeRequest =
        DescribeInstancesRequest.builder()
            .instanceIds(instanceId)
            .build();

    Ec2AsyncWaiter ec2Waiter = Ec2AsyncWaiter.builder()
        .client(getAsyncClient())
        .build();

    CompletableFuture<Void> resultFuture = new CompletableFuture<>();
    logger.info("Stopping instance " + instanceId + " and waiting for it to
stop.");
    getAsyncClient().stopInstances(stopRequest)
        .thenCompose(response -> {
            if (response.stoppingInstances().isEmpty()) {
                return CompletableFuture.failedFuture(new
RuntimeException("No instances were stopped. Please check the instance ID: " +
instanceId));
            }
        })
}
```

```

        return ec2Waiter.waitUntilInstanceStopped(describeRequest);
    })
    .thenAccept(waiterResponse -> {
        logger.info("Successfully stopped instance " + instanceId);
        resultFuture.complete(null);
    })
    .exceptionally(throwable -> {
        logger.error("Failed to stop instance " + instanceId + ": " +
            throwable.getMessage(), throwable);
        resultFuture.completeExceptionally(new RuntimeException("Failed
            to stop instance: " + throwable.getMessage(), throwable));
        return null;
    });

    return resultFuture;
}

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```

import { EC2Client, StopInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Stop one or more EC2 instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
    const client = new EC2Client({});
    const command = new StopInstancesCommand({
        InstanceIds: instanceIds,

```

```
});

try {
    const { StoppingInstances } = await client.send(command);
    const instanceIdList = StoppingInstances.map(
        (instance) => ` • ${instance.InstanceId}`,
    );
    console.log("Stopping instances:");
    console.log(instanceIdList.join("\n"));
} catch (caught) {
    if (
        caught instanceof Error &&
        caught.name === "InvalidInstanceID.NotFound"
    ) {
        console.warn(`${caught.message}`);
    } else {
        throw caught;
    }
}
};
```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```
suspend fun stopInstanceSc(instanceId: String) {
    val request =
        StopInstancesRequest {
            instanceIds = listOf(instanceId)
        }

    Ec2Client.fromEnvironment { region = "us-west-2" }.use { ec2 ->
        ec2.stopInstances(request)
    }
}
```

```

        println("Waiting until instance $instanceId stops. This will take a few
minutes.")
        ec2.waitForInstanceStopped {
            // suspend call
            instanceIds = listOf(instanceId)
        }
        println("Successfully stopped instance $instanceId")
    }
}

```

- 有关 API 的详细信息，请参阅适用[StopInstances](#)于 Kotlin 的 AWS SDK API 参考。

PowerShell

适用于 PowerShell V4 的工具

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 (V 4) [StopInstances](#) 中的。

适用于 PowerShell V5 的工具

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 (V 5) [StopInstances](#) 中的。

Python

适用于 Python 的 SDK (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。

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
        objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.
```



```

        :return: An instance of EC2InstanceWrapper initialized with the default
        EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)

    def stop(self) -> Optional[Dict[str, Any]]:
        """
        Stops instances and waits for them to be in a stopped state.

        :return: The response to the stop request, or None if there are no
        instances to stop.
        """
        if not self.instances:
            logger.info("No instances to stop.")
            return None

        instance_ids = [instance["InstanceId"] for instance in self.instances]
        try:
            # Attempt to stop the instances
            stop_response =
self.ec2_client.stop_instances(InstanceIds=instance_ids)
            waiter = self.ec2_client.get_waiter("instance_stopped")
            waiter.wait(InstanceIds=instance_ids)
        except ClientError as err:
            logger.error(
                f"Failed to stop instance(s): {' '.join(map(str, instance_ids))}"
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "IncorrectInstanceState":
                logger.error(
                    "Couldn't stop instance(s) because they are in an incorrect
state. "
                    "Ensure the instances are in a running state before stopping
them."
                )
            raise
        return stop_response

```

- 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](#) 中查找完整示例，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
end
```

```

end

ec2_client.stop_instances(instance_ids: [instance_id])
ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
puts 'Instance stopped.'
true
rescue StandardError => e
  puts "Error stopping instance: #{e.message}"
  false
end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-stop-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to stop instance '#{instance_id}' " \
    '(this might take a few minutes)... '
  return if instance_stopped?(ec2_client, instance_id)

  puts 'Could not stop instance.'
end

run_me if $PROGRAM_NAME == __FILE__

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```
pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error>
{
    tracing::info!("Stopping instance {instance_id}");

    self.client
        .stop_instances()
        .instance_ids(instance_id)
        .send()
        .await?;

    self.wait_for_instance_stopped(instance_id, None).await?;

    tracing::info!("Stopped instance.");

    Ok(())
}
```

使用等待器 API 等待实例进入已停止状态。使用等待器 API 要求在 rust 文件中添加“use aws_sdk_ec2::client::Waiter”。

```
pub async fn stop_instance(&self, instance_id: &str) -> Result<(), EC2Error>
{
    tracing::info!("Stopping instance {instance_id}");

    self.client
```

```

        .stop_instances()
        .instance_ids(instance_id)
        .send()
        .await?;

self.wait_for_instance_stopped(instance_id, None).await?;

tracing::info!("Stopped instance.");

Ok(())
}

```

- 有关 API 的详细信息，请参阅适用[StopInstances](#)于 Rust 的AWS SDK API 参考。

SAP ABAP

适用于 SAP ABAP 的 SDK

Note

还有更多相关信息 GitHub。在 [AWS 代码示例存储库](#) 中查找完整示例，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。

Swift

适用于 Swift 的 SDK

Note

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

```

import AWSEC2

/// Stop the specified instance.
///
/// - Parameters:
///   - instanceId: The ID of the instance to stop.
///   - waitUntilStopped: If `true`, execution waits until the instance
///     has stopped. Otherwise, execution continues and the instance stops
///     asynchronously.
///

```

```

    /// - Returns: `true` if the image is successfully stopped (or is left to
    ///   stop asynchronously). `false` if the instance doesn't stop.
    func stopInstance(instanceId: String, waitUntilStopped: Bool = false) async -
    > Bool {
        let instanceList = [instanceId]

        do {
            _ = try await ec2Client.stopInstances(
                input: StopInstancesInput(
                    instanceIds: instanceList
                )
            )

            if waitUntilStopped {
                print("Waiting for the instance to stop. Please be patient!")

                let waitOptions = WaiterOptions(maxWaitTime: 600)
                let output = try await ec2Client.waitUntilInstanceStopped(
                    options: waitOptions,
                    input: DescribeInstancesInput(
                        instanceIds: instanceList
                    )
                )

                switch output.result {
                case .success:
                    return true
                case .failure:
                    return false
                }
            } else {
                return true
            }
        } catch {
            print("*** Unable to stop the instance:
            \(error.localizedDescription)")
            return false
        }
    }
}

```

- 有关 API 的详细信息，请参阅适用于 Swift 的 AWS SDK API 参考 [StopInstances](#) 中。

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 SDK 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。您可以在以下代码示例中查看此操作的上下文：

- [了解基本功能](#)
- [开始使用 Amazon VPC](#)

.NET

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。

```
/// <summary>
/// Terminate an EC2 instance.
/// </summary>
/// <param name="ec2InstanceId">The instance Id of the EC2 instance
/// to terminate.</param>
/// <returns>Async task.</returns>
public async Task<List<InstanceStateChange>> TerminateInstances(string
ec2InstanceId)
{
    try
    {
        var request = new TerminateInstancesRequest
        {
            InstanceIds = new List<string> { ec2InstanceId }
        };

        var response = await _amazonEC2.TerminateInstancesAsync(request);
        Console.WriteLine("Waiting for the instance to terminate.");
    }
}
```



```

        await WaitForInstanceState(ec2InstanceId,
InstanceStateName.Terminated);

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

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

/// <summary>
/// Wait until an EC2 instance is in a specified state.
/// </summary>
/// <param name="instanceId">The instance Id.</param>
/// <param name="stateName">The state to wait for.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
{
    var request = new DescribeInstancesRequest
    {
        InstanceIds = new List<string> { instanceId }
    };

    // Wait until the instance is in the specified state.
    var hasState = false;
    do

```

```

    {
        // Wait 5 seconds.
        Thread.Sleep(5000);

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

    return hasState;
}

```

- 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](#) 中查找完整示例，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 getopt command in a function.

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

    # Retrieve the calling parameters.
    while getopt "i:h" option; do
        case "${option}" in
            i) instance_ids="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                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](#) 中查找完整示例，了解如何进行设置和运行。

```

//! Terminate an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
    \param instanceID: An EC2 instance ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::terminateInstances(const Aws::String &instanceID,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::TerminateInstancesRequest request;
    request.SetInstanceIds({instanceID});

    Aws::EC2::Model::TerminateInstancesOutcome outcome =
        ec2Client.TerminateInstances(request);
    if (outcome.IsSuccess()) {
        std::cout << "Ec2 instance '" << instanceID <<
            "' was terminated." << std::endl;
    }
}

```

```
    } else {
        std::cerr << "Failed to terminate ec2 instance " << instanceID <<
            ", " <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    return outcome.IsSuccess();
}
```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

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

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

```

    }).exceptionally(throwable -> {
        // Handle any exceptions that occurred during the async call
        throw new RuntimeException("Failed to terminate EC2 instance: " +
            throwable.getMessage(), throwable);
    });
}

```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```

import { EC2Client, TerminateInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Terminate one or more EC2 instances.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
    const client = new EC2Client({});
    const command = new TerminateInstancesCommand({
        InstanceIds: instanceIds,
    });

    try {
        const { TerminatingInstances } = await client.send(command);
        const instanceList = TerminatingInstances.map(
            (instance) => ` • ${instance.InstanceId}`,
        );
        console.log("Terminating instances:");
        console.log(instanceList.join("\n"));
    } catch (caught) {

```



```
    if (
        caught instanceof Error &&
        caught.name === "InvalidInstanceID.NotFound"
    ) {
        console.warn(`${caught.message}`);
    } else {
        throw caught;
    }
}
};
```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```
suspend fun terminateEC2(instanceID: String) {
    val request =
        TerminateInstancesRequest {
            instanceIds = listOf(instanceID)
        }

    Ec2Client.fromEnvironment { 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 Kotlin 的 AWS SDK API 参考。

PowerShell

适用于 PowerShell V4 的工具

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 会提示您进行确认；使用 -Force 开关可禁止提示。

```
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 \(V 4\)](#) [TerminateInstances](#) 中的。

适用于 PowerShell V5 的工具

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 会提示您进行确认；使用 -Force 开关可禁止提示。

```
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 \(V 5\)](#) [TerminateInstances](#) 中的。

Python

适用于 Python 的 SDK (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。

```
class EC2InstanceWrapper:
    """Encapsulates Amazon Elastic Compute Cloud (Amazon EC2) instance actions
    using the client interface."""

    def __init__(
        self, ec2_client: Any, instances: Optional[List[Dict[str, Any]]] = None
    ) -> None:
        """
        Initializes the EC2InstanceWrapper with an EC2 client and optional
        instances.

        :param ec2_client: A Boto3 Amazon EC2 client. This client provides low-
        level
                           access to AWS EC2 services.
        :param instances: A list of dictionaries representing Boto3 Instance
        objects. These are high-level objects that
                           wrap instance actions.
        """
        self.ec2_client = ec2_client
        self.instances = instances or []

    @classmethod
    def from_client(cls) -> "EC2InstanceWrapper":
        """
        Creates an EC2InstanceWrapper instance with a default EC2 client.

        :return: An instance of EC2InstanceWrapper initialized with the default
        EC2 client.
        """
        ec2_client = boto3.client("ec2")
        return cls(ec2_client)
```

```
def terminate(self) -> None:
    """
    Terminates instances and waits for them to reach the terminated state.
    """
    if not self.instances:
        logger.info("No instances to terminate.")
        return

    instance_ids = [instance["InstanceId"] for instance in self.instances]
    try:
        self.ec2_client.terminate_instances(InstanceIds=instance_ids)
        waiter = self.ec2_client.get_waiter("instance_terminated")
        waiter.wait(InstanceIds=instance_ids)
        self.instances.clear()
        for instance_id in instance_ids:
            print(f"• Instance ID: {instance_id}\n" f"• Action: Terminated")

    except ClientError as err:
        logger.error(
            f"Failed instance termination details:\n\t{str(self.instances)}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidInstanceID.NotFound":
            logger.error(
                "One or more instance IDs do not exist. "
                "Please verify the instance IDs and try again."
            )
        raise
```

- 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](#) 中查找完整示例，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.'
  true
rescue StandardError => e
  puts "Error terminating instance: #{e.message}"
  false
end
```

```

end

# Example usage:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    puts 'Example: ruby ec2-ruby-example-terminate-instance-i-123abc.rb ' \
      'i-123abc us-west-2'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)

  puts "Attempting to terminate instance '#{instance_id}' " \
    '(this might take a few minutes)... '
  return if instance_terminated?(ec2_client, instance_id)

  puts 'Could not terminate instance.'
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。

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](#) 中查找完整示例，了解如何进行设置和运行。

```
pub async fn delete_instance(&self, instance_id: &str) -> Result<(),
EC2Error> {
    tracing::info!("Deleting instance with id {instance_id}");
    self.stop_instance(instance_id).await?;
    self.client
        .terminate_instances()
        .instance_ids(instance_id)
        .send()
        .await?;
    self.wait_for_instance_terminated(instance_id).await?;
    tracing::info!("Terminated instance with id {instance_id}");
    Ok(())
}
```

使用等待器 API 等待实例进入已终止状态。使用等待器 API 要求在 rust 文件中添加“use aws_sdk_ec2::client::Waiter”。

```
async fn wait_for_instance_terminated(&self, instance_id: &str) -> Result<(),
EC2Error> {
    self.client
        .wait_until_instance_terminated()
        .instance_ids(instance_id)
        .wait(Duration::from_secs(60))
        .await
        .map_err(|err| match err {
            WaiterError::ExceededMaxWait(exceeded) => EC2Error(format!(
                "Exceeded max time ({}s) waiting for instance to terminate.",
                exceeded.max_wait().as_secs(),
            )),
            _ => EC2Error::from(err),
        })?;
}
```

```
    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用[TerminateInstances](#)于 Rust 的 AWS SDK API 参考。

Swift

适用于 Swift 的 SDK

Note

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

```
import AWSEC2

/// Terminate the specified instance.
///
/// - Parameters:
///   - instanceId: The instance to terminate.
///   - waitUntilTerminated: Whether or not to wait until the instance is
///     terminated before returning.
///
/// - Returns: `true` if terminated successfully. `false` if not or if an
///   error occurs.
func terminateInstance(instanceId: String, waitUntilTerminated: Bool = false)
async -> Bool {
    let instanceList = [instanceId]

    do {
        _ = try await ec2Client.terminateInstances(
            input: TerminateInstancesInput(
                instanceIds: instanceList
            )
        )

        if waitUntilTerminated {
            print("Waiting for the instance to terminate...")

            let waitOptions = WaiterOptions(maxWaitTime: 600.0)
```



```

        let output = try await ec2Client.waitForInstanceTerminated(
            options: waitOptions,
            input: DescribeInstancesInput(
                instanceIds: instanceList
            )
        )

        switch output.result {
        case .success:
            return true
        case .failure:
            return false
        }
    } else {
        return true
    }
} catch {
    print("*** Unable to terminate the instance:
\\(error.localizedDescription)")
    return false
}
}
}

```

- 有关 API 的详细信息，请参阅适用于 Swift 的 AWS SDK API 参考 [TerminateInstances](#) 中。

有关 AWS SDK 开发者指南和代码示例的完整列表，请参阅 [使用 AWS SDK 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。

将 `UnassignPrivateIpAddresses` 与 CLI 配合使用

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

CLI

AWS CLI

取消分配给网络接口的辅助私有 IP 地址

此示例取消分配给指定网络接口的指定私有 IP 地址。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

适用于 PowerShell V4 的工具

示例 1：此示例从指定的网络接口取消分配指定的私有 IP 地址。

```
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 (V 4) [UnassignPrivateIpAddresses](#)中的。

适用于 PowerShell V5 的工具

示例 1：此示例从指定的网络接口取消分配指定的私有 IP 地址。

```
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 (V 5) [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 SDK 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：

- [了解基本功能](#)

C++

SDK for C++

 Note

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

```

//! Disable monitoring for an EC2 instance.
/*!
    \param instanceId: An EC2 instance ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::disableMonitoring(const Aws::String &instanceId,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::UnmonitorInstancesRequest unrequest;
    unrequest.AddInstanceIds(instanceId);
    unrequest.SetDryRun(true);

    Aws::EC2::Model::UnmonitorInstancesOutcome dryRunOutcome =
    ec2Client.UnmonitorInstances(unrequest);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to disable monitoring on instance. A dry run
should trigger an error."
            <<
            std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
        Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to disable monitoring on instance " <<
            instanceId << ": " << dryRunOutcome.GetError().GetMessage() <<
            std::endl;
        return false;
    }

    unrequest.SetDryRun(false);

```

```
Aws::EC2::Model::UnmonitorInstancesOutcome unmonitorInstancesOutcome =
ec2Client.UnmonitorInstances(unrequest);
if (!unmonitorInstancesOutcome.IsSuccess()) {
    std::cout << "Failed to disable monitoring on instance " << instanceId
        << ": " << unmonitorInstancesOutcome.GetError().GetMessage() <<
        std::endl;
} else {
    std::cout << "Successfully disable monitoring on instance " <<
        instanceId << std::endl;
}

return unmonitorInstancesOutcome.IsSuccess();
}
```

- 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](#) 中查找完整示例，了解如何进行设置和运行。

```
import { EC2Client, UnmonitorInstancesCommand } from "@aws-sdk/client-ec2";
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";

/**
 * Turn off detailed monitoring for the selected instance.
 * @param {{ instanceIds: string[] }} options
 */
export const main = async ({ instanceIds }) => {
  const client = new EC2Client({});
  const command = new UnmonitorInstancesCommand({
    InstanceIds: instanceIds,
  });

  try {
    const { InstanceMonitorings } = await client.send(command);
    const instanceMonitoringsList = InstanceMonitorings.map(
      (im) =>
        ` • Detailed monitoring state for ${im.InstanceId} is
${im.Monitoring.State}.`,
    );
    console.log("Monitoring status:");
    console.log(instanceMonitoringsList.join("\n"));
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInstanceID.NotFound"
    ) {
      console.warn(`${caught.message}`);
    } else {

```

```
        throw caught;
    }
}
};
```

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例禁用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 (V 4) [UnmonitorInstances](#) 中的。

适用于 PowerShell V5 的工具

示例 1：此示例禁用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 (V 5) [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 SDK 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。

将 UpdateSecurityGroupRuleDescriptionsIngress 与 CLI 配合使用

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

CLI

AWS CLI

示例 1：使用 CIDR 源更新入站安全组规则的描述

以下 update-security-group-rule-descriptions-ingress 示例更新了指定端口和 IPv4 地址范围的安全组规则的描述。描述“SSH access from ABC office”取代该规则的任何现有描述。

```
aws ec2 update-security-group-rule-descriptions-ingress \
  --group-id sg-02f0d35a850ba727f \
  --ip-permissions
  IpProtocol=tcp,FromPort=22,ToPort=22,IpRanges='[{"CidrIp=203.0.113.0/16,Description="SSH
  access from corpnet"}]'
```

输出：

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

有关更多信息，请参阅 Amazon EC2 用户指南中的[安全组规则](#)。

示例 2：使用前缀列表源更新入站安全组规则的描述

以下 update-security-group-rule-descriptions-ingress 示例更新指定端口和前缀列表的安全组规则的描述。描述“SSH access from ABC office”取代该规则的任何现有描述。

```
aws ec2 update-security-group-rule-descriptions-ingress \
  --group-id sg-02f0d35a850ba727f \
  --ip-permissions
  IpProtocol=tcp,FromPort=22,ToPort=22,PrefixListIds='[{"PrefixListId=pl-12345678,Description="SSH
  access from corpnet"}]'
```

输出：

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

有关更多信息，请参阅 Amazon EC2 用户指南中的[安全组规则](#)。

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

PowerShell

适用于 PowerShell V4 的工具

示例 1：更新现有入口（入站）安全组规则的描述。

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupId
"sgr-1234567890"
$ruleWithUpdatedDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupId" = $existingInboundRule.SecurityGroupId
  "Description" = "Updated rule description"
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId
$existingInboundRule.GroupId -SecurityGroupRuleDescription
$ruleWithUpdatedDescription
```

示例 2：移除现有入口（入站）安全组规则的描述（通过在请求中省略该参数）。

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupId
"sgr-1234567890"
$ruleWithoutDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
  "SecurityGroupId" = $existingInboundRule.SecurityGroupId
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId
$existingInboundRule.GroupId -SecurityGroupRuleDescription
$ruleWithoutDescription
```


- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 4\) UpdateSecurityGroupRuleDescriptionsIngress](#) 中的。

适用于 PowerShell V5 的工具

示例 1：更新现有入口（入站）安全组规则的描述。

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupId
"sgr-1234567890"
$ruleWithUpdatedDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
    "SecurityGroupId" = $existingInboundRule.SecurityGroupId
    "Description" = "Updated rule description"
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId
$existingInboundRule.GroupId -SecurityGroupRuleDescription
$ruleWithUpdatedDescription
```

示例 2：移除现有入口（入站）安全组规则的描述（通过在请求中省略该参数）。

```
$existingInboundRule = Get-EC2SecurityGroupRule -SecurityGroupId
"sgr-1234567890"
$ruleWithoutDescription = [Amazon.EC2.Model.SecurityGroupRuleDescription]@{
    "SecurityGroupId" = $existingInboundRule.SecurityGroupId
}

Update-EC2SecurityGroupRuleIngressDescription -GroupId
$existingInboundRule.GroupId -SecurityGroupRuleDescription
$ruleWithoutDescription
```

- 有关 API 的详细信息，请参阅 [AWS Tools for PowerShell Cmdlet 参考 \(V 5\) UpdateSecurityGroupRuleDescriptionsIngress](#) 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 SDK 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。

Amazon EC2 使用的场景 AWS SDKs

以下代码示例向您展示了如何使用在 Amazon EC2 中实现常见场景 AWS SDKs。这些场景向您展示了如何通过调用 Amazon 中的多个函数 EC2 或其他函数结合使用来完成特定任务 AWS 服务。每个场景都包含完整源代码的链接，您可以在其中找到有关如何设置和运行代码的说明。

场景以中等水平的经验为目标，可帮助您结合具体环境了解服务操作。

示例

- [使用 AWS SDK 构建和管理弹性服务](#)
- [使用 CLI 创建具有私有子网和 NAT 网关的 VPC。](#)
- [开始通过 CLI 使用 Amazon VPC](#)
- [使用 CLI 的中转网关入门](#)
- [开始通过 CLI 使用 Amazon VPC IPAM](#)

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 A EC2 uto Scaling 组根据启动模板创建亚马逊弹性计算云 (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 网络服务器来处理 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

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.TargetGroup
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($"http://{endPoint}\n");
}
else
{
    Console.WriteLine(
        "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
        + "manually verifying that your VPC and security group are
configured correctly and that\n"
        + "you can successfully make a GET request to the load balancer
endpoint:\n");
    Console.WriteLine($"http://{endPoint}\n");
}
Console.WriteLine(new string('-', 80));

```

```

        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
            "to create situations where the web service fails, and
shows how using a resilient\n" +
            "architecture can keep the web service running in spite
of these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
            $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
            $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
        await
            _smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
                "this-is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +

```

```

        "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(

```

```

        $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
        "bad credentials...\n"
    );
    await _autoScalerWrapper.ReplaceInstanceProfile(
        badInstanceId,
        _autoScalerWrapper.BadCredsProfileName,
        instanceProfile.AssociationId
    );
    Console.WriteLine(
        "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
        "depending on which instance is selected by the load balancer.\n"
    );
    if (interactive)
        await DemoActionChoices();

    Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
    Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
    Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
    Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
    Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

    Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
    Console.WriteLine("and take that instance out of rotation.");

    await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

    Console.WriteLine($"Now, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
    Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
    Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
    Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");

```

```
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"Even while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");

        Console.WriteLine($"When all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
```

```

    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
            _elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
            _elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
            _autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
            _autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
            _autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
            );
            await
            _recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
            );
        }
    }

```

```

        Console.WriteLine(new string('-', 80));
        return true;
    }

```

创建一个包含 Auto Scaling 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 ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";

    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>

```

```

    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }

    /// <summary>
    /// Create a policy, role, and profile that is associated with instances with
    a specified name.
    /// An instance's associated profile defines a role that is assumed by the
    /// instance. The role has attached policies that specify the AWS permissions
    granted to
    /// clients that run on the instance.
    /// </summary>
    /// <param name="policyName">Name to use for the policy.</param>
    /// <param name="roleName">Name to use for the role.</param>
    /// <param name="profileName">Name to use for the profile.</param>
    /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>

```



```

    /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
    the role.</param>
    /// <returns>The Arn of the profile.</returns>
    public async Task<string> CreateInstanceProfileWithName(
        string policyName,
        string roleName,
        string profileName,
        string ssmOnlyPolicyFile,
        List<string>? awsManagedPolicies = null)
    {

        var assumeRoleDoc = "{" +
                                "\"Version\": \"2012-10-17\", " +
                                "\"Statement\": [{" +
                                    "\"Effect\": \"Allow\", " +
                                    "\"Principal\": {" +
                                    "\"Service\": [" +
                                        "\"ec2.amazonaws.com\"" +
                                    "]" +
                                    "}, " +
                                    "\"Action\": \"sts:AssumeRole\"" +
                                "}] " +
                                "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {

```

```

        Scope = PolicyScopeType.Local
    });
    // Get the entire list using the paginator.
    await foreach (var policy in policiesPaginator.Policies)
    {
        if (policy.PolicyName.Equals(policyName))
        {
            policyArn = policy.Arn;
        }
    }

    if (policyArn == null)
    {
        throw new InvalidOperationException("Policy not found");
    }
}

try
{
    await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
    {
        RoleName = roleName,
        AssumeRolePolicyDocument = assumeRoleDoc,
    });
    await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
    {
        RoleName = roleName,
        PolicyArn = policyArn
    });
    if (awsManagedPolicies != null)
    {
        foreach (var awsPolicy in awsManagedPolicies)
        {
            await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
            {
                PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                RoleName = roleName
            });
        }
    }
}
catch (EntityAlreadyExistsException)
{

```

```

        Console.WriteLine("Role already exists.");
    }

    string profileArn = "";
    try
    {
        var profileCreateResponse = await
            _amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
        // Allow time for the profile to be ready.
        profileArn = profileCreateResponse.InstanceProfile.Arn;
        Thread.Sleep(10000);
        await _amazonIam.AddRoleToInstanceProfileAsync(
            new AddRoleToInstanceProfileRequest()
            {
                InstanceProfileName = profileName,
                RoleName = roleName
            });
    }
    catch (EntityAlreadyExistsException)
    {
        Console.WriteLine("Policy already exists.");
        var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
            new GetInstanceProfileRequest()
            {
                InstanceProfileName = profileName
            });
        profileArn = profileGetResponse.InstanceProfile.Arn;
    }
    return profileArn;
}

/// <summary>
/// Create a new key pair and save the file.
/// </summary>
/// <param name="newKeyName">The name of the new key pair.</param>
/// <returns>Async task.</returns>
public async Task CreateKeyPair(string newKeyName)
{
    try

```

```

        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }

    /// <summary>
    /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
    Scaling.
    /// The launch template specifies a Bash script in its user data field that
    runs after
    /// the instance is started. This script installs the Python packages and
    starts a Python
    /// web server on the instance.
    /// </summary>
    /// <param name="startupScriptPath">The path to a Bash script file that is
    run.</param>

```

```

    /// <param name="instancePolicyPath">The path to a permissions policy to
    create and attach to the profile.</param>
    /// <returns>The template object.</returns>
    public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
    startupScriptPath, string instancePolicyPath)
    {
        try
        {
            await CreateKeyPair(_keyPairName);
            await CreateInstanceProfileWithName(_instancePolicyName,
            _instanceRoleName,
                _instanceProfileName, instancePolicyPath);

            var startServerText = await File.ReadAllTextAsync(startupScriptPath);
            var plainTextBytes =
            System.Text.Encoding.UTF8.GetBytes(startServerText);

            var amiLatest = await _amazonSsm.GetParameterAsync(
                new GetParameterRequest() { Name = _amiParam });
            var amiId = amiLatest.Parameter.Value;
            var launchTemplateResponse = await
            _amazonEc2.CreateLaunchTemplateAsync(
                new CreateLaunchTemplateRequest()
                {
                    LaunchTemplateName = _launchTemplateName,
                    LaunchTemplateData = new RequestLaunchTemplateData()
                    {
                        InstanceType = _instanceType,
                        ImageId = amiId,
                        IamInstanceProfile =
                            new
            LaunchTemplateIamInstanceProfileSpecificationRequest()
                            {
                                Name = _instanceProfileName
                            },
                        KeyName = _keyPairName,
                        UserData = System.Convert.ToBase64String(plainTextBytes)
                    }
                });
            return launchTemplateResponse.LaunchTemplate;
        }
        catch (AmazonEC2Exception ec2Exception)
        {

```

```

        if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
        {
            _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                $"Please try again with a unique name.");
        }

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

/// <summary>
/// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
/// </summary>
/// <returns>A list of availability zones.</returns>
public async Task<List<string>> DescribeAvailabilityZones()
{
    try
    {
        var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
            new DescribeAvailabilityZonesRequest());
        return zoneResponse.AvailabilityZones.Select(z =>
z.ZoneName).ToList();
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        _logger.LogError($"An Amazon EC2 error occurred while listing
availability zones.: {ec2Exception.Message}");
        throw;
    }
    catch (Exception ex)
    {
        _logger.LogError($"An error occurred while listing availability
zones.: {ex.Message}");
        throw;
    }
}

```

```

    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new
Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
size {groupSize}.");
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
exists.");
        }
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>

```

```

public async Task<Vpc> GetDefaultVpc()
{
    try
    {
        var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
            new DescribeVpcsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new("is-default", new List<string>() { "true" })
                }
            });
        return vpcResponse.Vpcs[0];
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "UnauthorizedOperation")
        {
            _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
        }

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

/// <summary>
/// Get all the subnets for a Vpc in a set of availability zones.
/// </summary>
/// <param name="vpcId">The Id of the Vpc.</param>
/// <param name="availabilityZones">The list of availability zones.</param>
/// <returns>The collection of subnet objects.</returns>
public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
{
    try
    {
        var subnets = new List<Subnet>();
    }
}

```



```

        var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
            new DescribeSubnetsRequest()
            {
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new("vpc-id", new List<string>() { vpcId }),
                    new("availability-zone", availabilityZones),
                    new("default-for-az", new List<string>() { "true" })
                }
            });

        // Get the entire list using the paginator.
        await foreach (var subnet in subnetPaginator.Subnets)
        {
            subnets.Add(subnet);
        }

        return subnets;
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
        {
            _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
        }

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

/// <summary>
/// Delete a launch template by name.
/// </summary>
/// <param name="templateName">The name of the template to delete.</param>
/// <returns>Async task.</returns>
public async Task DeleteTemplateByName(string templateName)
{

```

```

        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
                "InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name
{ _launchTemplateName } was not found.");
            }

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

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,

```

```

        RoleName = roleName
    });
    await _amazonIam.DeleteInstanceProfileAsync(
        new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
    var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
        new ListAttachedRolePoliciesRequest() { RoleName = roleName });
    foreach (var policy in attachedPolicies.AttachedPolicies)
    {
        await _amazonIam.DetachRolePolicyAsync(
            new DetachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policy.PolicyArn
            });
        // Delete the custom policies only.
        if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
        {
            await _amazonIam.DeletePolicyAsync(
                new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                {
                    PolicyArn = policy.PolicyArn
                });
        }
    }

    await _amazonIam.DeleteRoleAsync(
        new DeleteRoleRequest() { RoleName = roleName });
}
catch (NoSuchEntityException)
{
    Console.WriteLine($"Instance profile {profileName} does not exist.");
}
}

/// <summary>
/// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
/// </summary>
/// <param name="group">The name of the auto scaling group.</param>
/// <returns>A collection of instance Ids.</returns>
public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
{

```

```

        var instanceResponse = await
        _amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
            _amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {

```

```

        _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
        throw;
    }
}

/// <summary>
/// Replace the profile associated with a running instance. After the profile
is replaced, the instance
/// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
/// used to restart the Python web server.
/// </summary>
/// <param name="instanceId">The Id of the instance to update.</param>
/// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
/// <param name="associationId">The Id of the existing profile association
for the instance.</param>
/// <returns>Async task.</returns>
public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
{
    try
    {
        await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
            new ReplaceIamInstanceProfileAssociationRequest()
            {
                AssociationId = associationId,
                IamInstanceProfile = new IamInstanceProfileSpecification()
                {
                    Name = credsProfileName
                }
            });
        // Allow time before resetting.
        Thread.Sleep(25000);

        await _amazonEc2.RebootInstancesAsync(
            new RebootInstancesRequest(new List<string>() { instanceId }));
        Thread.Sleep(25000);
        var instanceReady = false;
        var retries = 5;
        while (retries-- > 0 && !instanceReady)
        {
            var instancesPaginator =

```

```

        _amazonSsm.Paginators.DescribeInstanceInformation(
            new DescribeInstanceInformationRequest());
        // Get the entire list using the paginator.
        await foreach (var instance in
instancesPaginator.InstanceInformationList)
        {
            instanceReady = instance.InstanceId == instanceId;
            if (instanceReady)
            {
                break;
            }
        }
        Console.WriteLine("Waiting for instance to be running.");
        await WaitForInstanceState(instanceId, InstanceStateName.Running);
        Console.WriteLine("Instance ready.");
        Console.WriteLine($"Sending restart command to instance
{instanceId}");
        await _amazonSsm.SendCommandAsync(
            new SendCommandRequest()
            {
                InstanceIds = new List<string>() { instanceId },
                DocumentName = "AWS-RunShellScript",
                Parameters = new Dictionary<string, List<string>>()
                {
                    {
                        "commands",
                        new List<string>() { "cd / && sudo python3 server.py
80" }
                    }
                }
            });
        Console.WriteLine($"Restarted the web server on instance
{instanceId}");
    }
    catch (AmazonEC2Exception ec2Exception)
    {
        if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
        {
            _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
        }

        throw;
    }

```

```

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

/// <summary>
/// Try to terminate an instance by its Id.
/// </summary>
/// <param name="instanceId">The Id of the instance to terminate.</param>
/// <returns>Async task.</returns>
public async Task TryTerminateInstanceById(string instanceId)
{
    var stopping = false;
    Console.WriteLine($"Stopping {instanceId}...");
    while (!stopping)
    {
        try
        {
            await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                new TerminateInstanceInAutoScalingGroupRequest()
                {
                    InstanceId = instanceId,
                    ShouldDecrementDesiredCapacity = false
                });
            stopping = true;
        }
        catch (ScalingActivityInProgressException)
        {
            Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
            Thread.Sleep(10000);
        }
    }
}

/// <summary>
/// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
/// waits and retries until the group is successfully deleted.

```

```

    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                    || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
        _amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {

```



```

        // Update the size to 0.
        await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
            new UpdateAutoScalingGroupRequest()
            {
                AutoScalingGroupName = groupName,
                MinSize = 0
            });
        var group = describeGroupsResponse.AutoScalingGroups[0];
        foreach (var instance in group.Instances)
        {
            await TryTerminateInstanceById(instance.InstanceId);
        }

        await TryDeleteGroupByName(groupName);
    }
    else
    {
        Console.WriteLine($"No groups found with name {groupName}.");
    }
}

/// <summary>
/// Get the default security group for a specified Vpc.
/// </summary>
/// <param name="vpc">The Vpc to search.</param>
/// <returns>The default security group.</returns>
public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
{
    var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
        new DescribeSecurityGroupsRequest()
        {
            Filters = new List<Amazon.EC2.Model.Filter>()
            {
                new ("group-name", new List<string>() { "default" }),
                new ("vpc-id", new List<string>() { vpc.VpcId })
            }
        });
    return groupResponse.SecurityGroups[0];
}

/// <summary>
/// Verify the default security group of a Vpc allows ingress from the
calling computer.

```

```

    /// This can be done by allowing ingress from this computer's IP address.
    /// In some situations, such as connecting from a corporate network, you must
    instead specify
    /// a prefix list Id. You can also temporarily open the port to any IP
    address while running this example.
    /// If you do, be sure to remove public access when you're done.
    /// </summary>
    /// <param name="vpc">The group to check.</param>
    /// <param name="port">The port to verify.</param>
    /// <param name="ipAddress">This computer's IP address.</param>
    /// <returns>True if the ip address is allowed on the group.</returns>
    public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
    {
        var portIsOpen = false;
        foreach (var ipPermission in group.IpPermissions)
        {
            if (ipPermission.FromPort == port)
            {
                foreach (var ipRange in ipPermission.Ipv4Ranges)
                {
                    var cidr = ipRange.CidrIp;
                    if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                    {
                        portIsOpen = true;
                    }
                }

                if (ipPermission.PrefixListIds.Any())
                {
                    portIsOpen = true;
                }

                if (!portIsOpen)
                {
                    Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                                "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
                }
                else
                {
                    break;
                }
            }
        }
    }

```

```

        }
    }

    return portIsOpen;
}

/// <summary>
/// Add an ingress rule to the specified security group that allows access on
the
/// specified port from the specified IP address.
/// </summary>
/// <param name="groupId">The Id of the security group to modify.</param>
/// <param name="port">The port to open.</param>
/// <param name="ipAddress">The IP address to allow access.</param>
/// <returns>Async task.</returns>
public async Task OpenInboundPort(string groupId, int port, string ipAddress)
{
    await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
        new AuthorizeSecurityGroupIngressRequest()
        {
            GroupId = groupId,
            IpPermissions = new List<IpPermission>()
            {
                new IpPermission()
                {
                    FromPort = port,
                    ToPort = port,
                    IpProtocol = "tcp",
                    Ipv4Ranges = new List<IpRange>()
                    {
                        new IpRange() { CidrIp = $"{ipAddress}/32" }
                    }
                }
            }
        });
}

/// <summary>
/// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
/// The
/// </summary>
/// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>

```

```

    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

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

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.WriteLine(". ");
        } while (!hasState);

        return hasState;
    }
}

```

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.WriteLine($"Creating table {tableName}...");
        var createRequest = new CreateTableRequest()
        {
            TableName = tableName,
            AttributeDefinitions = new List<AttributeDefinition>()
            {
                new AttributeDefinition()
                {
                    AttributeName = "MediaType",
                    AttributeType = ScalarAttributeType.S
                },
                new AttributeDefinition()
                {
                    AttributeName = "ItemId",
                    AttributeType = ScalarAttributeType.N
                }
            },
            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.WriteLine("\nWaiting for table to become active...");

        var request = new DescribeTableRequest
        {
            TableName = tableName
        };

        TableStatus status;
        do
        {
            Thread.Sleep(2000);

            var describeTableResponse = await
            _amazonDynamoDb.DescribeTableAsync(request);
            status = describeTableResponse.Table.TableStatus;

            Console.WriteLine(".");
        }
        while (status != "ACTIVE");

        return status == TableStatus.ACTIVE;
    }
    catch (ResourceInUseException)
    {
        Console.WriteLine($"Table {tableName} already exists.");
        return false;
    }
}

/// <summary>
/// Populate the database table with data from a specified path.
/// </summary>
/// <param name="databaseTableName">The name of the table.</param>
/// <param name="recommendationsPath">The path of the recommendations data.</
param>
/// <returns>Async task.</returns>
public async Task PopulateDatabase(string databaseTableName, string
recommendationsPath)
{
    var recommendationsText = await
File.ReadAllTextAsync(recommendationsPath);
    var records =

JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);

```

```

        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}

```

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 Reference》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 wasSuccessful =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessful) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                        For this example to work, the default security group
for your default VPC must
                        allow access from this computer. You can either add
it automatically from this

```

```

        example or add it yourself using the AWS Management
        Console.

        """);

        System.out.println(
            "Do you want to add a rule to security group " +
            groupInfo.getGroupName() + " to allow");
        System.out.println("inbound traffic on port " + port + " from
        your computer's IP address (y/n) ");
        String ans = in.nextLine();
        if ("y".equalsIgnoreCase(ans)) {
            autoScaler.openInboundPort(groupInfo.getGroupName(),
            String.valueOf(port), ipAddress);
            System.out.println("Security group rule added.");
        } else {
            System.out.println("No security group rule added.");
        }
    }

    } catch (AutoScalingException e) {
        e.printStackTrace();
    }
} else if (wasSuccessful) {
    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 =
HttpClientClients.createDefault();

                    // Create an HTTP GET request to the ELB.
                    HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                    // Execute the request and get the response.
                    HttpResponse response = httpClient.execute(httpGet);
                    int statusCode =
response.getStatusLine().getStatusCode();
                    System.out.println("HTTP Status Code: " + statusCode);

                    // Display the JSON response
                    BufferedReader reader = new BufferedReader(
                        new
InputStreamReader(response.getEntity().getContent()));
                    StringBuilder jsonResponse = new StringBuilder();
                    String line;
                    while ((line = reader.readLine()) != null) {
                        jsonResponse.append(line);
                    }
                    reader.close();

                    // Print the formatted JSON response.
                    System.out.println("Full Response:\n");
                    System.out.println(jsonResponse.toString());
```

```

        // Close the HTTP client.
        httpClient.close();

    }
    case 1 -> {
        System.out.println("\nChecking the health of load
balancer targets:\n");
        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
        for (TargetHealthDescription target : health) {
            System.out.printf("\tTarget %s on port %d is %s\n",
target.target().id(),
                                target.target().port(),
target.targetHealth().stateAsString());
        }
        System.out.println("""
                                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 Auto Scaling 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.getInstanceInformationList();
        for (InstanceInformation info : instanceInformationList) {
            if (info.getInstanceId().equals(instanceId)) {
                instReady = true;
                break;
            }
        }
    }

```

```

        }
    }
}

SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
    .instanceIds(instanceId)
    .documentName("AWS-RunShellScript")
    .parameters(Collections.singletonMap("commands",
        Collections.singletonList("cd / && sudo python3 server.py
80")))
    .build();

getSSMClient().sendCommand(sendCommandRequest);
System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
}

public void openInboundPort(String secGroupId, String port, String ipAddress)
{
    AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
        .groupName(secGroupId)
        .cidrIp(ipAddress)
        .fromPort(Integer.parseInt(port))
        .build();

    getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
    System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
}

/**
 * Detaches a role from an instance profile, detaches policies from the role,
 * and deletes all the resources.
 */
public void deleteInstanceProfile(String roleName, String profileName) {
    try {
        software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
        .builder()
        .instanceProfileName(profileName)
        .build();
    }
}

```

```

        GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
        String name = response.getInstanceProfile().getInstanceProfileName();
        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.getAttachedPolicies();
        for (AttachedPolicy attachedPolicy : attachedPolicies) {
            DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
                .roleName(roleName)
                .policyArn(attachedPolicy.getPolicyArn())
                .build();

            getIAMClient().detachRolePolicy(request);
            System.out.println("Detached and deleted policy " +
attachedPolicy.getPolicyName());
        }

        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)
        .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 Reference》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](#)

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 "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes] [-v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}

```

创建部署所有资源的步骤。

```

import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";

import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,

```

```

    DescribeSecurityGroupsCommand,
    AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
    IAMClient,
    CreatePolicyCommand,
    CreateRoleCommand,
    CreateInstanceProfileCommand,
    AddRoleToInstanceProfileCommand,
    AttachRolePolicyCommand,
    waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
    CreateAutoScalingGroupCommand,
    AutoScalingClient,
    AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
    CreateListenerCommand,
    CreateLoadBalancerCommand,
    CreateTargetGroupCommand,
    ElasticLoadBalancingV2Client,
    waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
    ScenarioOutput,
    ScenarioInput,
    ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "../constants.js";
import { initParamsSteps } from "../steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
    new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
    new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
        type: "confirm",
    })
];

```



```

    }),
    new ScenarioAction(
        "handleConfirmDeployment",
        (c) => c.confirmDeployment === false && process.exit(),
    ),
    new ScenarioOutput(
        "creatingTable",
        MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
    ),
    new ScenarioAction("createTable", async () => {
        const client = new DynamoDBClient({});
        await client.send(
            new CreateTableCommand({
                TableName: NAMES.tableName,
                ProvisionedThroughput: {
                    ReadCapacityUnits: 5,
                    WriteCapacityUnits: 5,
                },
                AttributeDefinitions: [
                    {
                        AttributeName: "MediaType",
                        AttributeType: "S",
                    },
                    {
                        AttributeName: "ItemId",
                        AttributeType: "N",
                    },
                ],
                KeySchema: [
                    {
                        AttributeName: "MediaType",
                        KeyType: "HASH",
                    },
                    {
                        AttributeName: "ItemId",
                        KeyType: "RANGE",
                    },
                ],
            })
        );
        await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
    }),
    new ScenarioOutput(
        "createdTable",

```

```

    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
    */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),

```

```

new ScenarioOutput(
  "createdKeyPair",
  MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
),
new ScenarioOutput(
  "creatingInstancePolicy",
  MESSAGES.creatingInstancePolicy.replace(
    "${INSTANCE_POLICY_NAME}",
    NAMES.instancePolicyName,
  ),
),
new ScenarioAction("createInstancePolicy", async (state) => {
  const client = new IAMClient({});
  const {
    Policy: { Arn },
  } = await client.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.instancePolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "instance_policy.json"),
      ),
    }),
  ),
  state.instancePolicyArn = Arn;
}),
new ScenarioOutput("createdInstancePolicy", (state) =>
  MESSAGES.createdInstancePolicy
    .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
    .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
),
new ScenarioOutput(
  "creatingInstanceRole",
  MESSAGES.creatingInstanceRole.replace(
    "${INSTANCE_ROLE_NAME}",
    NAMES.instanceRoleName,
  ),
),
new ScenarioAction("createInstanceRole", () => {
  const client = new IAMClient({});
  return client.send(
    new CreateRoleCommand({
      RoleName: NAMES.instanceRoleName,
      AssumeRolePolicyDocument: readFileSync(
        join(ROOT, "assume-role-policy.json"),
      ),
    }),
  ),
});

```

```

    ),
  })),
);
}),
new ScenarioOutput(
  "createdInstanceRole",
  MESSAGES.createdInstanceRole.replace(
    "${INSTANCE_ROLE_NAME}",
    NAMES.instanceRoleName,
  ),
),
new ScenarioOutput(
  "attachingPolicyToRole",
  MESSAGES.attachingPolicyToRole
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
    .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
),
new ScenarioAction("attachPolicyToRole", async (state) => {
  const client = new IAMClient({});
  await client.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.instanceRoleName,
      PolicyArn: state.instancePolicyArn,
    }),
  );
}),
new ScenarioOutput(
  "attachedPolicyToRole",
  MESSAGES.attachedPolicyToRole
    .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
),
new ScenarioOutput(
  "creatingInstanceProfile",
  MESSAGES.creatingInstanceProfile.replace(
    "${INSTANCE_PROFILE_NAME}",
    NAMES.instanceProfileName,
  ),
),
new ScenarioAction("createInstanceProfile", async (state) => {
  const client = new IAMClient({});
  const {
    InstanceProfile: { Arn },
  } = await client.send(

```

```

        new CreateInstanceProfileCommand({
            InstanceProfileName: NAMES.instanceProfileName,
        }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
        { client },
        { InstanceProfileName: NAMES.instanceProfileName },
    );
}),
new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
),
new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
),
new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
        new AddRoleToInstanceProfileCommand({
            RoleName: NAMES.instanceRoleName,
            InstanceProfileName: NAMES.instanceProfileName,
        }),
    );
}),
new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
),
...initParamsSteps,
new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
    const { Parameter } = await ssmClient.send(
        new GetParameterCommand({
            Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",

```

```

    })),
  );
const ec2Client = new EC2Client({});
await ec2Client.send(
  new CreateLaunchTemplateCommand({
    LaunchTemplateName: NAMES.launchTemplateName,
    LaunchTemplateData: {
      InstanceType: "t3.micro",
      ImageId: Parameter.Value,
      IamInstanceProfile: { Name: NAMES.instanceProfileName },
      UserData: readFileSync(
        join(RESOURCES_PATH, "server_startup_script.sh"),
      ).toString("base64"),
      KeyName: NAMES.keyPairName,
    },
  })),
);
}),
new ScenarioOutput(
  "createdLaunchTemplate",
  MESSAGES.createdLaunchTemplate.replace(
    "${LAUNCH_TEMPLATE_NAME}",
    NAMES.launchTemplateName,
  ),
),
new ScenarioOutput(
  "creatingAutoScalingGroup",
  MESSAGES.creatingAutoScalingGroup.replace(
    "${AUTO_SCALING_GROUP_NAME}",
    NAMES.autoScalingGroupName,
  ),
),
new ScenarioAction("createAutoScalingGroup", async (state) => {
  const ec2Client = new EC2Client({});
  const { AvailabilityZones } = await ec2Client.send(
    new DescribeAvailabilityZonesCommand({}),
  );
  state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
  const autoScalingClient = new AutoScalingClient({});
  await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
    autoScalingClient.send(
      new CreateAutoScalingGroupCommand({
        AvailabilityZones: state.availabilityZoneNames,
        AutoScalingGroupName: NAMES.autoScalingGroupName,

```

```

        LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
        },
        MinSize: 3,
        MaxSize: 3,
    )),
),
);
}),
new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
        MESSAGES.createdAutoScalingGroup
            .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
            .replace(
                "${AVAILABILITY_ZONE_NAMES}",
                state.availabilityZoneNames.join(", "),
            ),
),
new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
}),
new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
        new DescribeVpcsCommand({
            Filters: [{ Name: "is-default", Values: ["true"] }],
        }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
}),
new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
),
new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(

```

```

    new DescribeSubnetsCommand({
      Filters: [
        { Name: "vpc-id", Values: [state.defaultVpc] },
        { Name: "availability-zone", Values: state.availabilityZoneNames },
        { Name: "default-for-az", Values: ["true"] },
      ],
    }),
  );
  state.subnets = Subnets.map((subnet) => subnet.SubnetId);
}),
new ScenarioOutput(
  "gotSubnets",
  /**
   * @param {{ subnets: string[] }} state
   */
  (state) =>
    MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
),
new ScenarioOutput(
  "creatingLoadBalancerTargetGroup",
  MESSAGES.creatingLoadBalancerTargetGroup.replace(
    "${TARGET_GROUP_NAME}",
    NAMES.loadBalancerTargetGroupName,
  ),
),
new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new CreateTargetGroupCommand({
      Name: NAMES.loadBalancerTargetGroupName,
      Protocol: "HTTP",
      Port: 80,
      HealthCheckPath: "/healthcheck",
      HealthCheckIntervalSeconds: 10,
      HealthCheckTimeoutSeconds: 5,
      HealthyThresholdCount: 2,
      UnhealthyThresholdCount: 2,
      VpcId: state.defaultVpc,
    }),
  );
  const targetGroup = TargetGroups[0];
  state.targetGroupArn = targetGroup.TargetGroupArn;
  state.targetGroupProtocol = targetGroup.Protocol;
  state.targetGroupPort = targetGroup.Port;
});

```



```

   )),
    new ScenarioOutput(
      "createdLoadBalancerTargetGroup",
      MESSAGES.createdLoadBalancerTargetGroup.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      ),
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,

```

```

        Port: state.targetGroupPort,
        DefaultActions: [
            { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
    )),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  )),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    ),
  );
  )),
  new ScenarioOutput(
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-ec2').SecurityGroup}} state
     */
    async (state) => {
      const client = new EC2Client({});

```

```

const { SecurityGroups } = await client.send(
  new DescribeSecurityGroupsCommand({
    Filters: [{ Name: "group-name", Values: ["default"] }],
  }),
);
if (!SecurityGroups) {
  state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
}
state.defaultSecurityGroup = SecurityGroups[0];

/**
 * @type {string}
 */
const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
state.myIp = ipResponse.trim();
const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
  ({ IpRanges }) =>
    IpRanges.some(
      ({ CidrIp }) =>
        CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
    ),
)
  .filter(({ IpProtocol }) => IpProtocol === "tcp")
  .filter(({ FromPort }) => FromPort === 80);

state.myIpRules = myIpRules;
},
),
new ScenarioOutput(
  "verifiedInboundPort",
  /**
   * @param {{ myIpRules: any[] }} state
   */
  (state) => {
    if (state.myIpRules.length > 0) {
      return MESSAGES.foundIpRules.replace(
        "${IP_RULES}",
        JSON.stringify(state.myIpRules, null, 2),
      );
    }
    return MESSAGES.noIpRules;
  },
),
new ScenarioInput(

```

```

    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
        if (state.myIpRules.length > 0) {
            return false;
        }
        return MESSAGES.noIpRules;
    },
    { type: "confirm" },
),
new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-ec2').SecurityGroup }} state
     */
    async (state) => {
        if (!state.shouldAddInboundRule) {
            return;
        }

        const client = new EC2Client({});
        await client.send(
            new AuthorizeSecurityGroupIngressCommand({
                GroupId: state.defaultSecurityGroup.GroupId,
                CidrIp: `${state.myIp}/32`,
                FromPort: 80,
                ToPort: 80,
                IpProtocol: "tcp",
            }),
        );
    },
),
new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
        return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
}),
new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
),

```

```

new ScenarioAction("verifyEndpoint", async (state) => {
  try {
    const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
      axios.get(`http://${state.loadBalancerDns}`),
    );
    state.endpointResponse = JSON.stringify(response.data, null, 2);
  } catch (e) {
    state.verifyEndpointError = e;
  }
}),
new ScenarioOutput("verifiedEndpoint", (state) => {
  if (state.verifyEndpointError) {
    console.error(state.verifyEndpointError);
  } else {
    return MESSAGES.verifiedEndpoint.replace(
      "${ENDPOINT_RESPONSE}",
      state.endpointResponse,
    );
  }
}),
saveState,
];

```

创建运行演示的步骤。

```

import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";

import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {

```

```

    IAMClient,
    CreatePolicyCommand,
    CreateRoleCommand,
    AttachRolePolicyCommand,
    CreateInstanceProfileCommand,
    AddRoleToInstanceProfileCommand,
    waitUntilInstanceProfileExists,
  } from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {

```

```

        throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
},
);

const getRecommendationResult = new ScenarioOutput(
    "getRecommendationResult",
    (state) =>
        `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
    { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
            Names: [NAMES.loadBalancerTargetGroupName],
        }),
    );
    const { TargetHealthDescriptions } = await client.send(
        new DescribeTargetHealthCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
        }),
    );
    state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
    "getHealthCheckResult",
    /**
     * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
    balancing-v2').TargetHealthDescription[]}} state
     */
    (state) => {
        const status = state.targetHealthDescriptions
            .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
            .join("\n");
        return `Health check:\n${status}`;
    },
    { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(

```

```

    "loadBalancerLoop",
    getRecommendation.action,
    {
      whileConfig: {
        whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
        input: new ScenarioInput(
          "loadBalancerCheck",
          MESSAGES.demoLoadBalancerCheck,
          {
            type: "confirm",
          },
        ),
        output: getRecommendationResult,
      },
    },
  );

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);

const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),

```



```

...statusSteps,
new ScenarioInput(
  "brokenDependencyConfirmation",
  MESSAGES.demoBrokenDependencyConfirmation,
  { type: "confirm" },
),
new ScenarioAction("brokenDependency", async (state) => {
  if (!state.brokenDependencyConfirmation) {
    process.exit();
  } else {
    const client = new SSMClient({});
    state.badTableName = `fake-table-${Date.now()}`;
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: state.badTableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }
}),
new ScenarioOutput("testBrokenDependency", (state) =>
  MESSAGES.demoTestBrokenDependency.replace(
    "${TABLE_NAME}",
    state.badTableName,
  ),
),
...statusSteps,
new ScenarioInput(
  "staticResponseConfirmation",
  MESSAGES.demoStaticResponseConfirmation,
  { type: "confirm" },
),
new ScenarioAction("staticResponse", async (state) => {
  if (!state.staticResponseConfirmation) {
    process.exit();
  } else {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmFailureResponseKey,
        Value: "static",
        Overwrite: true,

```

```

        Type: "String",
    )),
    );
}
)),
new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
...statusSteps,
new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
),
new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
        process.exit();
    }
}),
new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
        new PutParameterCommand({
            Name: NAMES.ssmTableNameKey,
            Value: NAMES.tableName,
            Overwrite: true,
            Type: "String",
        )),
    );
}),
new ScenarioAction(
    "badCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
     */
    async (state) => {
        await createSsmOnlyInstanceProfile();
        const autoScalingClient = new AutoScalingClient({});
        const { AutoScalingGroups } = await autoScalingClient.send(
            new DescribeAutoScalingGroupsCommand({
                AutoScalingGroupNames: [NAMES.autoScalingGroupName],
            )),
        );
        state.targetInstance = AutoScalingGroups[0].Instances[0];
        const ec2Client = new EC2Client({});

```

```

const { IamInstanceProfileAssociations } = await ec2Client.send(
  new DescribeIamInstanceProfileAssociationsCommand({
    Filters: [
      { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
    ],
  }),
);
state.instanceProfileAssociationId =
  IamInstanceProfileAssociations[0].AssociationId;
await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
  ec2Client.send(
    new ReplaceIamInstanceProfileAssociationCommand({
      AssociationId: state.instanceProfileAssociationId,
      IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
    }),
  ),
);

await ec2Client.send(
  new RebootInstancesCommand({
    InstanceIds: [state.targetInstance.InstanceId],
  }),
);

const ssmClient = new SSMClient({});
await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
  const { InstanceInformationList } = await ssmClient.send(
    new DescribeInstanceInformationCommand({}),
  );

  const instance = InstanceInformationList.find(
    (info) => info.InstanceId === state.targetInstance.InstanceId,
  );

  if (!instance) {
    throw new Error("Instance not found.");
  }
});

await ssmClient.send(
  new SendCommandCommand({
    InstanceIds: [state.targetInstance.InstanceId],
    DocumentName: "AWS-RunShellScript",
    Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
  }),
);

```

```

        })),
    );
  },
),
new ScenarioOutput(
  "testBadCredentials",
  /**
   * @param {{ targetInstance: import('@aws-sdk/client-
   ssm').InstanceInformation}} state
   */
  (state) =>
    MESSAGES.demoTestBadCredentials.replace(
      "${INSTANCE_ID}",
      state.targetInstance.InstanceId,
    ),
),
loadBalancerLoop,
new ScenarioInput(
  "deepHealthCheckConfirmation",
  MESSAGES.demoDeepHealthCheckConfirmation,
  { type: "confirm" },
),
new ScenarioAction("deepHealthCheckExit", (state) => {
  if (!state.deepHealthCheckConfirmation) {
    process.exit();
  }
})),
new ScenarioAction("deepHealthCheck", async () => {
  const client = new SSMClient({});
  await client.send(
    new PutParameterCommand({
      Name: NAMES.ssmHealthCheckKey,
      Value: "deep",
      Overwrite: true,
      Type: "String",
    }),
  ),
);
}),
new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
healthCheckLoop,
loadBalancerLoop,
new ScenarioInput(
  "killInstanceConfirmation",
  /**

```

```

    * @param {{ targetInstance: import('@aws-sdk/client-
    ssm').InstanceInformation }} state
    */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
      { type: "confirm" },
    ),
    new ScenarioAction("killInstanceExit", (state) => {
      if (!state.killInstanceConfirmation) {
        process.exit();
      }
    }),
    new ScenarioAction(
      "killInstance",
      /**
       * @param {{ targetInstance: import('@aws-sdk/client-
       ssm').InstanceInformation }} state
       */
      async (state) => {
        const client = new AutoScalingClient({});
        await client.send(
          new TerminateInstanceInAutoScalingGroupCommand({
            InstanceId: state.targetInstance.InstanceId,
            ShouldDecrementDesiredCapacity: false,
          }),
        );
      },
    ),
    new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
    healthCheckLoop,
    loadBalancerLoop,
    new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
      type: "confirm",
    }),
    new ScenarioAction("failOpenExit", (state) => {
      if (!state.failOpenConfirmation) {
        process.exit();
      }
    }),
    new ScenarioAction("failOpen", () => {
      const client = new SSMClient({});

```

```

    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  })),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  })),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  })),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(

```

```

        join(RESOURCES_PATH, "ssm_only_policy.json"),
    ),
  )),
);
await iamClient.send(
  new CreateRoleCommand({
    RoleName: NAMES.ssmOnlyRoleName,
    AssumeRolePolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Principal: { Service: "ec2.amazonaws.com" },
          Action: "sts:AssumeRole",
        },
      ],
    }),
  )),
);
await iamClient.send(
  new AttachRolePolicyCommand({
    RoleName: NAMES.ssmOnlyRoleName,
    PolicyArn: Policy.Arn,
  )),
);
await iamClient.send(
  new AttachRolePolicyCommand({
    RoleName: NAMES.ssmOnlyRoleName,
    PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
  )),
);
const { InstanceProfile } = await iamClient.send(
  new CreateInstanceProfileCommand({
    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
  )),
);
await waitUntilInstanceProfileExists(
  { client: iamClient },
  { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
);
await iamClient.send(
  new AddRoleToInstanceProfileCommand({
    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    RoleName: NAMES.ssmOnlyRoleName,
  }
);

```

```

    }),
  );

  return InstanceProfile;
}

```

创建销毁所有资源的步骤。

```

import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,

```



```

    ScenarioInput,
    ScenarioAction,
  } from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "../constants.js";
import { findLoadBalancer } from "../shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    }
  })
];

```

```

    } catch (e) {
      state.deleteKeyPairError = e;
    }
  )),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  )),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
          })
        );
      }
    } catch (e) {
      state.detachPolicyFromRoleError = e;
    }
  )),
  new ScenarioOutput("detachedPolicyFromRole", (state) => {
    if (state.detachPolicyFromRoleError) {
      console.error(state.detachPolicyFromRoleError);
      return MESSAGES.detachPolicyFromRoleError
        .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
  })
);

```

```

    }
    return MESSAGES.detachedPolicyFromRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  })),
  new ScenarioAction("deleteInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const policy = await findPolicy(NAMES.instancePolicyName);

    if (!policy) {
      state.deletePolicyError = new Error(
        `Policy ${NAMES.instancePolicyName} not found.`
      );
    } else {
      return client.send(
        new DeletePolicyCommand({
          PolicyArn: policy.Arn,
        }),
      );
    }
  })),
  new ScenarioOutput("deletePolicyResult", (state) => {
    if (state.deletePolicyError) {
      console.error(state.deletePolicyError);
      return MESSAGES.deletePolicyError.replace(
        "${INSTANCE_POLICY_NAME}",
        NAMES.instancePolicyName,
      );
    }
    return MESSAGES.deletedPolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  })),
  new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          RoleName: NAMES.instanceRoleName,
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {

```

```

        state.removeRoleFromInstanceProfileError = e;
    }
  })),
  new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
    if (state.removeRoleFromInstanceProfile) {
      console.error(state.removeRoleFromInstanceProfileError);
      return MESSAGES.removeRoleFromInstanceProfileError
        .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
        .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
    }
    return MESSAGES.removedRoleFromInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  })),
  new ScenarioAction("deleteInstanceRole", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteRoleCommand({
          RoleName: NAMES.instanceRoleName,
        }),
      );
    } catch (e) {
      state.deleteInstanceRoleError = e;
    }
  })),
  new ScenarioOutput("deleteInstanceRoleResult", (state) => {
    if (state.deleteInstanceRoleError) {
      console.error(state.deleteInstanceRoleError);
      return MESSAGES.deleteInstanceRoleError.replace(
        "${INSTANCE_ROLE_NAME}",
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  })),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({

```

```

        InstanceProfileName: NAMES.instanceProfileName,
    )),
    );
} catch (e) {
    state.deleteInstanceProfileError = e;
}
}),
new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
        console.error(state.deleteInstanceProfileError);
        return MESSAGES.deleteInstanceProfileError.replace(
            "${INSTANCE_PROFILE_NAME}",
            NAMES.instanceProfileName,
        );
    }
    return MESSAGES.deletedInstanceProfile.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
    );
}),
new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
        await terminateGroupInstances(NAMES.autoScalingGroupName);
        await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
            await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
        });
    } catch (e) {
        state.deleteAutoScalingGroupError = e;
    }
}),
new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
    if (state.deleteAutoScalingGroupError) {
        console.error(state.deleteAutoScalingGroupError);
        return MESSAGES.deleteAutoScalingGroupError.replace(
            "${AUTO_SCALING_GROUP_NAME}",
            NAMES.autoScalingGroupName,
        );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
    );
}),
new ScenarioAction("deleteLaunchTemplate", async (state) => {

```

```

const client = new EC2Client({});
try {
  await client.send(
    new DeleteLaunchTemplateCommand({
      LaunchTemplateName: NAMES.launchTemplateName,
    }),
  );
} catch (e) {
  state.deleteLaunchTemplateError = e;
}
}),
new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
  if (state.deleteLaunchTemplateError) {
    console.error(state.deleteLaunchTemplateError);
    return MESSAGES.deleteLaunchTemplateError.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }
  return MESSAGES.deletedLaunchTemplate.replace(
    "${LAUNCH_TEMPLATE_NAME}",
    NAMES.launchTemplateName,
  );
}),
new ScenarioAction("deleteLoadBalancer", async (state) => {
  try {
    const client = new ElasticLoadBalancingV2Client({});
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    await client.send(
      new DeleteLoadBalancerCommand({
        LoadBalancerArn: loadBalancer.LoadBalancerArn,
      }),
    );
    await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
      const lb = await findLoadBalancer(NAMES.loadBalancerName);
      if (lb) {
        throw new Error("Load balancer still exists.");
      }
    });
  } catch (e) {
    state.deleteLoadBalancerError = e;
  }
}),
new ScenarioOutput("deleteLoadBalancerResult", (state) => {

```

```

    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  })),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
  })),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  });

```

```

    })),
    new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
      try {
        const client = new IAMClient({});
        await client.send(
          new RemoveRoleFromInstanceProfileCommand({
            InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
            RoleName: NAMES.ssmOnlyRoleName,
          }),
        );
      } catch (e) {
        state.detachSsmOnlyRoleFromProfileError = e;
      }
    })),
    new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
      if (state.detachSsmOnlyRoleFromProfileError) {
        console.error(state.detachSsmOnlyRoleFromProfileError);
        return MESSAGES.detachSsmOnlyRoleFromProfileError
          .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
          .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
      }
      return MESSAGES.detachedSsmOnlyRoleFromProfile
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    })),
    new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
      try {
        const iamClient = new IAMClient({});
        const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
        await iamClient.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.ssmOnlyRoleName,
            PolicyArn: ssmOnlyPolicy.Arn,
          }),
        );
      } catch (e) {
        state.detachSsmOnlyCustomRolePolicyError = e;
      }
    })),
    new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
      if (state.detachSsmOnlyCustomRolePolicyError) {
        console.error(state.detachSsmOnlyCustomRolePolicyError);
        return MESSAGES.detachSsmOnlyCustomRolePolicyError
          .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)

```



```

        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    })),
    new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
        try {
            const iamClient = new IAMClient({});
            await iamClient.send(
                new DetachRolePolicyCommand({
                    RoleName: NAMES.ssmOnlyRoleName,
                    PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
                }),
            );
        } catch (e) {
            state.detachSsmOnlyAWSRolePolicyError = e;
        }
    })),
    new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
        if (state.detachSsmOnlyAWSRolePolicyError) {
            console.error(state.detachSsmOnlyAWSRolePolicyError);
            return MESSAGES.detachSsmOnlyAWSRolePolicyError
                .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
                .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
        }
        return MESSAGES.detachedSsmOnlyAWSRolePolicy
            .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
            .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    })),
    new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
        try {
            const iamClient = new IAMClient({});
            await iamClient.send(
                new DeleteInstanceProfileCommand({
                    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
                }),
            );
        } catch (e) {
            state.deleteSsmOnlyInstanceProfileError = e;
        }
    })),
    new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
        if (state.deleteSsmOnlyInstanceProfileError) {

```

```

        console.error(state.deleteSsmOnlyInstanceProfileError);
        return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
            "${INSTANCE_PROFILE_NAME}",
            NAMES.ssmOnlyInstanceProfileName,
        );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
    );
}),
new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
        const iamClient = new IAMClient({});
        const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
        await iamClient.send(
            new DeletePolicyCommand({
                PolicyArn: ssmOnlyPolicy.Arn,
            }),
        );
    } catch (e) {
        state.deleteSsmOnlyPolicyError = e;
    }
}),
new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
        console.error(state.deleteSsmOnlyPolicyError);
        return MESSAGES.deleteSsmOnlyPolicyError.replace(
            "${POLICY_NAME}",
            NAMES.ssmOnlyPolicyName,
        );
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
    );
}),
new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
        const iamClient = new IAMClient({});
        await iamClient.send(
            new DeleteRoleCommand({
                RoleName: NAMES.ssmOnlyRoleName,
            }),
        );
    }

```

```

    );
  } catch (e) {
    state.deleteSsmOnlyRoleError = e;
  }
}),
new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
  if (state.deleteSsmOnlyRoleError) {
    console.error(state.deleteSsmOnlyRoleError);
    return MESSAGES.deleteSsmOnlyRoleError.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }
  return MESSAGES.deletedSsmOnlyRole.replace(
    "${ROLE_NAME}",
    NAMES.ssmOnlyRoleName,
  );
}),
new ScenarioAction(
  "revokeSecurityGroupIngress",
  async (
    /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
    state,
  ) => {
    const ec2Client = new EC2Client({});

    try {
      await ec2Client.send(
        new RevokeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
        }),
      );
    } catch (e) {
      state.revokeSecurityGroupIngressError = e;
    }
  },
),
new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
  if (state.revokeSecurityGroupIngressError) {
    console.error(state.revokeSecurityGroupIngressError);
  }
}

```

```

        return MESSAGES.revokeSecurityGroupIngressError.replace(
            "${IP}",
            state.myIp,
        );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  })),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
}

/**

```

```

    * @param {string} groupName
    */
    async function terminateGroupInstances(groupName) {
        const autoScalingClient = new AutoScalingClient({});
        const group = await findAutoScalingGroup(groupName);
        await autoScalingClient.send(
            new UpdateAutoScalingGroupCommand({
                AutoScalingGroupName: group.AutoScalingGroupName,
                MinSize: 0,
            }),
        );
        for (const i of group.Instances) {
            await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
                autoScalingClient.send(
                    new TerminateInstanceInAutoScalingGroupCommand({
                        InstanceId: i.InstanceId,
                        ShouldDecrementDesiredCapacity: true,
                    }),
                ),
            );
        }
    }

    async function findAutoScalingGroup(groupName) {
        const client = new AutoScalingClient({});
        const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
        for await (const page of paginatedGroups) {
            const group = page.AutoScalingGroups.find(
                (g) => g.AutoScalingGroupName === groupName,
            );
            if (group) {
                return group;
            }
        }
        throw new Error(`Auto scaling group ${groupName} not found.`);
    }

```

- 有关 API 详细信息，请参阅《AWS SDK for JavaScript API Reference》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

适用于 Python 的 SDK (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:
    """
    Manages the deployment, demonstration, and destruction of resources for the
    resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example,
        such as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
        DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
        """
```

```
"""
recommendations_path = f"{self.resource_path}/recommendations.json"
startup_script = f"{self.resource_path}/server_startup_script.sh"
instance_policy = f"{self.resource_path}/instance_policy.json"

logging.info("Starting deployment of resources for the resilient
service.")

logging.info(
    "Creating and populating DynamoDB table '%s'.",
    self.recommendation.table_name,
)
self.recommendation.create()
self.recommendation.populate(recommendations_path)

logging.info(
    "Creating an EC2 launch template with the startup script '%s'.",
    startup_script,
)
self.autoscaler.create_template(startup_script, instance_policy)

logging.info(
    "Creating an EC2 Auto Scaling group across multiple Availability
Zones."
)
zones = self.autoscaler.create_autoscaling_group(3)

logging.info("Creating variables that control the flow of the demo.")
self.param_helper.reset()

logging.info("Creating Elastic Load Balancing target group and load
balancer.")

vpc = self.autoscaler.get_default_vpc()
subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
target_group = self.loadbalancer.create_target_group(
    self.target_group_name, self.protocol, self.port, vpc["VpcId"]
)
self.loadbalancer.create_load_balancer(
    self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
)
self.loadbalancer.create_listener(self.load_balancer_name, target_group)

self.autoscaler.attach_load_balancer_target_group(target_group)
```



```

logging.info("Verifying access to the load balancer endpoint.")
endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

if not lb_success:
    logging.warning(
        "Couldn't connect to the load balancer. Verifying that the port
is open..."
    )
    sec_group, port_is_open = self.autoscaler.verify_inbound_port(
        vpc, self.port, current_ip_address
    )
    sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
        vpc, self.ssh_port, current_ip_address
    )
    if not port_is_open:
        logging.warning(
            "The default security group for your VPC must allow access
from this computer."
        )
        if q.ask(
            f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
            f"inbound traffic on port {self.port} from your computer's IP
address of {current_ip_address}? (y/n) ",
            q.is_yesno,
        ):
            self.autoscaler.open_inbound_port(
                sec_group["GroupId"], self.port, current_ip_address
            )
    if not ssh_port_is_open:
        if q.ask(
            f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
            f"inbound SSH traffic on port {self.ssh_port} for debugging
from your computer's IP address of {current_ip_address}? (y/n) ",
            q.is_yesno,
        ):
            self.autoscaler.open_inbound_port(
                sec_group["GroupId"], self.ssh_port, current_ip_address
            )

```

```

        lb_success =
self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer
endpoint. Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as
        sending requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
                logging.info("Sending a GET request to the load balancer
endpoint.")
                endpoint =
self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:

```

```

        state = target["TargetHealth"]["State"]
        logging.info(
            "Target %s on port %d is %s",
            target["Target"]["Id"],
            target["Target"]["Port"],
            state,
        )
        if state != "healthy":
            logging.warning(
                "%s: %s",
                target["TargetHealth"]["Reason"],
                target["TargetHealth"]["Description"],
            )
        logging.info(
            "Note that it can take a minute or two for the health check
to update."
        )
    elif choice == 2:
        logging.info("Proceeding to the next part of the demo.")

def demo(self) -> None:
    """
    Runs the demonstration, showing how the service responds to different
failure scenarios
and how a resilient architecture can keep the service running.
    """
    ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

    logging.info("Resetting parameters to starting values for the demo.")
    self.param_helper.reset()

    logging.info(
        "Starting demonstration of the service's resilience under various
failure conditions."
    )
    self.demo_choices()

    logging.info(
        "Simulating failure by changing the Systems Manager parameter to a
non-existent table."
    )
    self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
    logging.info("Sending GET requests will now return failure codes.")
    self.demo_choices()

```

```
logging.info("Switching to static response mode to mitigate failure.")
self.param_helper.put(self.param_helper.failure_response, "static")
logging.info("Sending GET requests will now return static responses.")
self.demo_choices()

logging.info("Restoring normal operation of the recommendation service.")
self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

logging.info(
    "Introducing a failure by assigning bad credentials to one of the
instances."
)
self.autoscaler.create_instance_profile(
    ssm_only_policy,
    self.autoscaler.bad_creds_policy_name,
    self.autoscaler.bad_creds_role_name,
    self.autoscaler.bad_creds_profile_name,
    ["AmazonSSMManagedInstanceCore"],
)
instances = self.autoscaler.get_instances()
bad_instance_id = instances[0]
instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
logging.info(
    "Replacing instance profile with bad credentials for instance %s.",
    bad_instance_id,
)
self.autoscaler.replace_instance_profile(
    bad_instance_id,
    self.autoscaler.bad_creds_profile_name,
    instance_profile["AssociationId"],
)
logging.info(
    "Sending GET requests may return either a valid recommendation or a
static response."
)
self.demo_choices()

logging.info("Implementing deep health checks to detect unhealthy
instances.")
self.param_helper.put(self.param_helper.health_check, "deep")
logging.info("Checking the health of the load balancer targets.")
self.demo_choices()
```

```

        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace
it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance
replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation
service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
        if automation:
            cleanup = True
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()

```

```

        self.autoscaler.delete_instance_profile(
            self.autoscaler.bad_creds_profile_name,
            self.autoscaler.bad_creds_role_name,
        )
        logging.info("Deleting DynamoDB table and other resources.")
        self.recommendation.destroy()
    else:
        logging.warning(
            "Resources have not been deleted. Ensure you clean them up
manually to avoid unexpected charges."
        )

def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the
demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../../scenarios/features/resilient_service/resources",
        help="The path to resource files used by this example, such as IAM
policies and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")

```

```

ec2_client = boto3.client("ec2")
ssm_client = boto3.client("ssm")
iam_client = boto3.client("iam")

# Wrapper instantiations
recommendation = RecommendationService(
    "doc-example-recommendation-service", ddb_client
)
autoscaling_wrapper = AutoScalingWrapper(
    prefix,
    "t3.micro",
    "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
    autoscaling_client,
    ec2_client,
    ssm_client,
    iam_client,
)
elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
param_helper = ParameterHelper(recommendation.table_name, ssm_client)

# Demo invocation
runner = Runner(
    args.resource_path,
    recommendation,
    autoscaling_wrapper,
    elb_wrapper,
    param_helper,
)
actions = [args.action] if args.action != "all" else ["deploy", "demo",
"destroy"]
for action in actions:
    if action == "deploy":
        runner.deploy()
    elif action == "demo":
        runner.demo()
    elif action == "destroy":
        runner.destroy()

logging.info("Demo completed successfully.")

if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()

```

创建一个包含 Auto Scaling 和 Amazon EC2 操作的类。

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

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
        created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
        that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
```



```

self.group_name = f"{resource_prefix}-group"

# Happy path
self.instance_policy_name = f"{resource_prefix}-pol"
self.instance_role_name = f"{resource_prefix}-role"
self.instance_profile_name = f"{resource_prefix}-prof"

# Failure mode
self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
self.bad_creds_role_name = f"{resource_prefix}-bc-role"
self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"

def create_policy(self, policy_file: str, policy_name: str) -> str:
    """
    Creates a new IAM policy or retrieves the ARN of an existing policy.

    :param policy_file: The path to a JSON file that contains the policy
    definition.
    :param policy_name: The name to give the created policy.
    :return: The ARN of the created or existing policy.
    """
    with open(policy_file) as file:
        policy_doc = file.read()

    try:
        response = self.iam_client.create_policy(
            PolicyName=policy_name, PolicyDocument=policy_doc
        )
        policy_arn = response["Policy"]["Arn"]
        log.info(f"Policy '{policy_name}' created successfully. ARN:
{policy_arn}")
        return policy_arn

    except ClientError as err:
        if err.response["Error"]["Code"] == "EntityAlreadyExists":
            # If the policy already exists, get its ARN
            response = self.iam_client.get_policy(
                PolicyArn=f"arn:aws:iam::{self.account_id}:policy/
{policy_name}"
            )
            policy_arn = response["Policy"]["Arn"]
            log.info(f"Policy '{policy_name}' already exists. ARN:
{policy_arn}")

```

```

        return policy_arn
    log.error(f"Full error:\n\t{err}")

def create_role(self, role_name: str, assume_role_doc: dict) -> str:
    """
    Creates a new IAM role or retrieves the ARN of an existing role.

    :param role_name: The name to give the created role.
    :param assume_role_doc: The assume role policy document that specifies
which
                           entities can assume the role.
    :return: The ARN of the created or existing role.
    """
    try:
        response = self.iam_client.create_role(
            RoleName=role_name,
AssumeRolePolicyDocument=json.dumps(assume_role_doc)
        )
        role_arn = response["Role"]["Arn"]
        log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
        return role_arn

    except ClientError as err:
        if err.response["Error"]["Code"] == "EntityAlreadyExists":
            # If the role already exists, get its ARN
            response = self.iam_client.get_role(RoleName=role_name)
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
            return role_arn
        log.error(f"Full error:\n\t{err}")

def attach_policy(
    self,
    role_name: str,
    policy_arn: str,
    aws_managed_policies: Tuple[str, ...] = (),
) -> None:
    """
    Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

    :param role_name: The name of the role to attach the policy to.
    :param policy_arn: The ARN of the policy to attach.

```

```

        :param aws_managed_policies: A tuple of AWS-managed policy names to
attach to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role
{role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is
assumed by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
            the role, such as
AmazonSSMManagedInstanceCore to grant

```

use of Systems Manager to send commands to

the instance.

```

        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

    try:
        profile_response = self.iam_client.create_instance_profile(
            InstanceProfileName=profile_name
        )
        waiter = self.iam_client.get_waiter("instance_profile_exists")
        waiter.wait(InstanceProfileName=profile_name)
        time.sleep(10) # wait a little longer
        profile_arn = profile_response["InstanceProfile"]["Arn"]
        self.iam_client.add_role_to_instance_profile(
            InstanceProfileName=profile_name, RoleName=role_name
        )
        log.info("Created profile %s and added role %s.", profile_name,
role_name)
    except ClientError as err:
        if err.response["Error"]["Code"] == "EntityAlreadyExists":
            prof_response = self.iam_client.get_instance_profile(
                InstanceProfileName=profile_name
            )
            profile_arn = prof_response["InstanceProfile"]["Arn"]
            log.info(
                "Instance profile %s already exists, nothing to do.",
profile_name
            )
            log.error(f"Full error:\n\t{err}")
        return profile_arn

```

```

def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
    """
    Gets data about the profile associated with an instance.

    :param instance_id: The ID of the instance to look up.
    :return: The profile data.
    """
    try:
        response =
self.ec2_client.describe_iam_instance_profile_associations(
            Filters=[{"Name": "instance-id", "Values": [instance_id]}]
        )
        if not response["IamInstanceProfileAssociations"]:
            log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
{instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")

def replace_instance_profile(
    self,
    instance_id: str,
    new_instance_profile_name: str,
    profile_association_id: str,
) -> None:
    """
    Replaces the profile associated with a running instance. After the
profile is
    replaced, the instance is rebooted to ensure that it uses the new
profile. When
    the instance is ready, Systems Manager is used to restart the Python web
server.

```

```

        :param instance_id: The ID of the instance to restart.
        :param new_instance_profile_name: The name of the new profile to
associate with
                                     the specified instance.
        :param profile_association_id: The ID of the existing profile association
for the
                                     instance.
    """
    try:
        self.ec2_client.replace_iam_instance_profile_association(
            IamInstanceProfile={"Name": new_instance_profile_name},
            AssociationId=profile_association_id,
        )
        log.info(
            "Replaced instance profile for association %s with profile %s.",
            profile_association_id,
            new_instance_profile_name,
        )
        time.sleep(5)

        self.ec2_client.reboot_instances(InstanceIds=[instance_id])
        log.info("Rebooting instance %s.", instance_id)
        waiter = self.ec2_client.get_waiter("instance_running")
        log.info("Waiting for instance %s to be running.", instance_id)
        waiter.wait(InstanceIds=[instance_id])
        log.info("Instance %s is now running.", instance_id)

        self.ssm_client.send_command(
            InstanceIds=[instance_id],
            DocumentName="AWS-RunShellScript",
            Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
        )
        log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
    except ClientError as err:
        log.error("Failed to replace instance profile.")
        error_code = err.response["Error"]["Code"]
        if error_code == "InvalidAssociationID.NotFound":
            log.error(
                f"Association ID '{profile_association_id}' does not exist."
                "Please check the association ID and try again."
            )
        if error_code == "InvalidInstanceId":
            log.error(

```

```

        f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
        f"Please verify the instance ID and try again."
    )
    log.error(f"Full error:\n\t{err}")

def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
    """
    Detaches a role from an instance profile, detaches policies from the
role,
and deletes all the resources.

:param profile_name: The name of the profile to delete.
:param role_name: The name of the role to delete.
    """
    try:
        self.iam_client.remove_role_from_instance_profile(
            InstanceProfileName=profile_name, RoleName=role_name
        )

    self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
    log.info("Deleted instance profile %s.", profile_name)
    attached_policies = self.iam_client.list_attached_role_policies(
        RoleName=role_name
    )
    for pol in attached_policies["AttachedPolicies"]:
        self.iam_client.detach_role_policy(
            RoleName=role_name, PolicyArn=pol["PolicyArn"]
        )
        if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
            self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
            log.info("Detached and deleted policy %s.", pol["PolicyName"])
    self.iam_client.delete_role(RoleName=role_name)
    log.info("Deleted role %s.", role_name)
except ClientError as err:
    log.error(
        f"Couldn't delete instance profile {profile_name} or detach "
        f"policies and delete role {role_name}: {err}"
    )
    if err.response["Error"]["Code"] == "NoSuchEntity":
        log.info(
            "Instance profile %s doesn't exist, nothing to do.",
profile_name

```

```

    )

def create_key_pair(self, key_pair_name: str) -> None:
    """
    Creates a new key pair.

    :param key_pair_name: The name of the key pair to create.
    """
    try:
        response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
        with open(f"{key_pair_name}.pem", "w") as file:
            file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
        log.info("Created key pair %s.", key_pair_name)
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(f"Failed to create key pair {key_pair_name}.")
        if error_code == "InvalidKeyPair.Duplicate":
            log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
        log.error(f"Full error:\n\t{err}")

def delete_key_pair(self) -> None:
    """
    Deletes a key pair.
    """
    try:
        self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
        remove(f"{self.key_pair_name}.pem")
        log.info("Deleted key pair %s.", self.key_pair_name)
    except ClientError as err:
        log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
        log.error(f"Full error:\n\t{err}")
    except FileNotFoundError as err:
        log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
        log.error(f"Full error:\n\t{err}")

def create_template(
    self, server_startup_script_file: str, instance_policy_file: str
) -> Dict[str, Any]:

```



```

        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                         when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
                                   to create and attach to the instance
profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )

            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                }
            )

```

```

        "UserData": base64.b64encode(
            start_server_script.encode(encoding="utf-8")
        ).decode(encoding="utf-8"),
        "KeyName": self.key_pair_name,
    },
)
template = lt_response["LaunchTemplate"]
log.info(
    f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
)
except ClientError as err:
    log.error(f"Failed to create launch template
{self.launch_template_name}.")
    error_code = err.response["Error"]["Code"]
    if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
        log.info(
            f"Launch template {self.launch_template_name} already exists,
nothing to do."
        )
    log.error(f"Full error:\n\t{err}")
return template

def delete_template(self):
    """
    Deletes a launch template.
    """
    try:
        self.ec2_client.delete_launch_template(
            LaunchTemplateName=self.launch_template_name
        )
        self.delete_instance_profile(
            self.instance_profile_name, self.instance_role_name
        )
        log.info("Launch template %s deleted.", self.launch_template_name)
    except ClientError as err:
        if (
            err.response["Error"]["Code"]
            == "InvalidLaunchTemplateName.NotFoundException"
        ):
            log.info(
                "Launch template %s does not exist, nothing to do.",
                self.launch_template_name,
            )

```

```

        )
        log.error(f"Full error:\n\t{err}")

def get_availability_zones(self) -> List[str]:
    """
    Gets a list of Availability Zones in the AWS Region of the Amazon EC2
    client.

    :return: The list of Availability Zones for the client Region.
    """
    try:
        response = self.ec2_client.describe_availability_zones()
        zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
        log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
    except ClientError as err:
        log.error("Failed to retrieve availability zones.")
        log.error(f"Full error:\n\t{err}")
    else:
        return zones

def create_autoscaling_group(self, group_size: int) -> List[str]:
    """
    Creates an EC2 Auto Scaling group with the specified size.

    :param group_size: The number of instances to set for the minimum and
    maximum in
                        the group.
    :return: The list of Availability Zones specified for the group.
    """
    try:
        zones = self.get_availability_zones()
        self.autoscaling_client.create_auto_scaling_group(
            AutoScalingGroupName=self.group_name,
            AvailabilityZones=zones,
            LaunchTemplate={
                "LaunchTemplateName": self.launch_template_name,
                "Version": "$Default",
            },
            MinSize=group_size,
            MaxSize=group_size,
        )
        log.info(

```

```

        f"Created EC2 Auto Scaling group {self.group_name} with
availability zones {zones}."
    )
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        if error_code == "AlreadyExists":
            log.info(
                f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
            )
        else:
            log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
            log.error(f"Full error:\n\t{err}")
    else:
        return zones

def get_instances(self) -> List[str]:
    """
    Gets data about the instances in the EC2 Auto Scaling group.

    :return: A list of instance IDs in the Auto Scaling group.
    """
    try:
        as_response = self.autoscaling_client.describe_auto_scaling_groups(
            AutoScalingGroupNames=[self.group_name]
        )
        instance_ids = [
            i["InstanceId"]
            for i in as_response["AutoScalingGroups"][0]["Instances"]
        ]
        log.info(
            f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
        )
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(
            f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
        )
        if error_code == "ResourceNotFound":

```

```

        log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
        log.error(f"Full error:\n\t{err}")
    else:
        return instance_ids

def terminate_instance(self, instance_id: str, decrementsetting=False) ->
None:
    """
    Terminates an instance in an EC2 Auto Scaling group. After an instance is
    terminated, it can no longer be accessed.

    :param instance_id: The ID of the instance to terminate.
    :param decrementsetting: If True, do not replace terminated instances.
    """
    try:
        self.autoscaling_client.terminate_instance_in_auto_scaling_group(
            InstanceId=instance_id,
            ShouldDecrementDesiredCapacity=decrementsetting,
        )
        log.info("Terminated instance %s.", instance_id)

        # Adding a waiter to ensure the instance is terminated
        waiter = self.ec2_client.get_waiter("instance_terminated")
        log.info("Waiting for instance %s to be terminated...", instance_id)
        waiter.wait(InstanceIds=[instance_id])
        log.info(
            f"Instance '{instance_id}' has been terminated and will be
replaced."
        )

    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(f"Failed to terminate instance '{instance_id}'.")
        if error_code == "ScalingActivityInProgressFault":
            log.error(
                "Scaling activity is currently in progress. "
                "Wait for the scaling activity to complete before attempting
to terminate the instance again."
            )
        elif error_code == "ResourceContentionFault":
            log.error(
                "The request failed due to a resource contention issue. "

```

```

        "Ensure that no conflicting operations are being performed on
the resource."
    )
    log.error(f"Full error:\n\t{err}")

def attach_load_balancer_target_group(
    self, lb_target_group: Dict[str, Any]
) -> None:
    """
    Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
    The target group specifies how the load balancer forwards requests to the
instances
    in the group.

    :param lb_target_group: Data about the ELB target group to attach.
    """
    try:
        self.autoscaling_client.attach_load_balancer_target_groups(
            AutoScalingGroupName=self.group_name,
            TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
        )
        log.info(
            "Attached load balancer target group %s to auto scaling group
%s.",
            lb_target_group["TargetGroupName"],
            self.group_name,
        )
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(
            f"Failed to attach load balancer target group
'{lb_target_group['TargetGroupName']}'."
        )
        if error_code == "ResourceContentionFault":
            log.error(
                "The request failed due to a resource contention issue. "
                "Ensure that no conflicting operations are being performed on
the resource."
            )
        elif error_code == "ServiceLinkedRoleFailure":
            log.error(
                "The operation failed because the service-linked role is not
ready or does not exist. "
            )

```

```

        "Check that the service-linked role exists and is correctly
configured."
    )
    log.error(f"Full error:\n\t{err}")

def delete_autoscaling_group(self, group_name: str) -> None:
    """
    Terminates all instances in the group, then deletes the EC2 Auto Scaling
group.

:param group_name: The name of the group to delete.
    """
    try:
        response = self.autoscaling_client.describe_auto_scaling_groups(
            AutoScalingGroupNames=[group_name]
        )
        groups = response.get("AutoScalingGroups", [])
        if len(groups) > 0:
            self.autoscaling_client.update_auto_scaling_group(
                AutoScalingGroupName=group_name, MinSize=0
            )
            instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
            for inst_id in instance_ids:
                self.terminate_instance(inst_id)

            # Wait for all instances to be terminated
            if instance_ids:
                waiter = self.ec2_client.get_waiter("instance_terminated")
                log.info("Waiting for all instances to be terminated...")
                waiter.wait(InstanceIds=instance_ids)
                log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
        if error_code == "ScalingActivityInProgressFault":
            log.error(
                "Scaling activity is currently in progress. "
                "Wait for the scaling activity to complete before attempting
to delete the group again."
            )

```

```

        elif error_code == "ResourceContentionFault":
            log.error(
                "The request failed due to a resource contention issue. "
                "Ensure that no conflicting operations are being performed on
the group."
            )
            log.error(f"Full error:\n\t{err}")

def get_default_vpc(self) -> Dict[str, Any]:
    """
    Gets the default VPC for the account.

    :return: Data about the default VPC.
    """
    try:
        response = self.ec2_client.describe_vpcs(
            Filters=[{"Name": "is-default", "Values": ["true"]}])
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error("Failed to retrieve the default VPC.")
        if error_code == "UnauthorizedOperation":
            log.error(
                "You do not have the necessary permissions to describe VPCs.
"
                "Ensure that your AWS IAM user or role has the correct
permissions."
            )
        elif error_code == "InvalidParameterValue":
            log.error(
                "One or more parameters are invalid. Check the request
parameters."
            )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass

```



```

def verify_inbound_port(
    self, vpc: Dict[str, Any], port: int, ip_address: str
) -> Tuple[Dict[str, Any], bool]:
    """
    Verify the default security group of the specified VPC allows ingress
    from this
    computer. This can be done by allowing ingress from this computer's IP
    address. In some situations, such as connecting from a corporate network,
    you
    must instead specify a prefix list ID. You can also temporarily open the
    port to
    any IP address while running this example. If you do, be sure to remove
    public
    access when you're done.

    :param vpc: The VPC used by this example.
    :param port: The port to verify.
    :param ip_address: This computer's IP address.
    :return: The default security group of the specified VPC, and a value
    that indicates
        whether the specified port is open.
    """
    try:
        response = self.ec2_client.describe_security_groups(
            Filters=[
                {"Name": "group-name", "Values": ["default"]},
                {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
            ]
        )
        sec_group = response["SecurityGroups"][0]
        port_is_open = False
        log.info(f"Found default security group {sec_group['GroupId']}.")

        for ip_perm in sec_group["IpPermissions"]:
            if ip_perm.get("FromPort", 0) == port:
                log.info(f"Found inbound rule: {ip_perm}")
                for ip_range in ip_perm["IpRanges"]:
                    cidr = ip_range.get("CidrIp", "")
                    if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                        port_is_open = True
                if ip_perm["PrefixListIds"]:
                    port_is_open = True
        if not port_is_open:

```

```

        log.info(
            f"The inbound rule does not appear to be open to
either this computer's IP "
            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
        )
    else:
        break
except ClientError as err:
    error_code = err.response["Error"]["Code"]
    log.error(
        f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
    )
    if error_code == "InvalidVpcID.NotFound":
        log.error(
            f"The specified VPC ID '{vpc['VpcId']}' does not exist.
Please check the VPC ID."
        )
        log.error(f"Full error:\n\t{err}")
    else:
        return sec_group, port_is_open

def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
    """
    Add an ingress rule to the specified security group that allows access on
the
    specified port from the specified IP address.

    :param sec_group_id: The ID of the security group to modify.
    :param port: The port to open.
    :param ip_address: The IP address that is granted access.
    """
    try:
        self.ec2_client.authorize_security_group_ingress(
            GroupId=sec_group_id,
            CidrIp=f"{ip_address}/32",
            FromPort=port,
            ToPort=port,
            IpProtocol="tcp",
        )
        log.info(

```

```

        "Authorized ingress to %s on port %s from %s.",
        sec_group_id,
        port,
        ip_address,
    )
except ClientError as err:
    error_code = err.response["Error"]["Code"]
    log.error(
        f"Failed to authorize ingress to security group '{sec_group_id}'
on port {port} from {ip_address}."
    )
    if error_code == "InvalidGroupId.Malformed":
        log.error(
            "The security group ID is malformed. "
            "Please verify that the security group ID is correct."
        )
    elif error_code == "InvalidPermission.Duplicate":
        log.error(
            "The specified rule already exists in the security group. "
            "Check the existing rules for this security group."
        )
    log.error(f"Full error:\n\t{err}")

def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
    """
    Gets the default subnets in a VPC for a specified list of Availability
    Zones.

    :param vpc_id: The ID of the VPC to look up.
    :param zones: The list of Availability Zones to look up.
    :return: The list of subnets found.
    """
    # Ensure that 'zones' is a list, even if None is passed
    if zones is None:
        zones = []
    try:
        paginator = self.ec2_client.get_paginator("describe_subnets")
        page_iterator = paginator.paginate(
            Filters=[
                {"Name": "vpc-id", "Values": [vpc_id]},
                {"Name": "availability-zone", "Values": zones},
                {"Name": "default-for-az", "Values": ["true"]},
            ]
        )

```

```

        ]
    )

    subnets = []
    for page in page_iterator:
        subnets.extend(page["Subnets"])

    log.info("Found %s subnets for the specified zones.", len(subnets))
    return subnets
except ClientError as err:
    log.error(
        f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
{zones}."
    )
    error_code = err.response["Error"]["Code"]
    if error_code == "InvalidVpcID.NotFound":
        log.error(
            "The specified VPC ID does not exist. "
            "Please check the VPC ID and try again."
        )
    # Add more error-specific handling as needed
    log.error(f"Full error:\n\t{err}")

```

创建一个包含弹性负载均衡操作的类。

```

class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client

    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """

```

Creates an Elastic Load Balancing target group. The target group specifies how the load balancer forwards requests to instances in the group and how instance health is checked.

To speed up this demo, the health check is configured with shortened times and lower thresholds. In production, you might want to decrease the sensitivity of your health checks to avoid unwanted failures.

```
:param target_group_name: The name of the target group to create.
:param protocol: The protocol to use to forward requests, such as 'HTTP'.
:param port: The port to use to forward requests, such as 80.
:param vpc_id: The ID of the VPC in which the load balancer exists.
:return: Data about the newly created target group.
"""
try:
    response = self.elb_client.create_target_group(
        Name=target_group_name,
        Protocol=protocol,
        Port=port,
        HealthCheckPath="/healthcheck",
        HealthCheckIntervalSeconds=10,
        HealthCheckTimeoutSeconds=5,
        HealthyThresholdCount=2,
        UnhealthyThresholdCount=2,
        VpcId=vpc_id,
    )
    target_group = response["TargetGroups"][0]
    log.info(f"Created load balancing target group
'{target_group_name}'.")
    return target_group
except ClientError as err:
    log.error(
        f"Couldn't create load balancing target group
'{target_group_name}'."
    )
    error_code = err.response["Error"]["Code"]

    if error_code == "DuplicateTargetGroupName":
        log.error(
            f"Target group name {target_group_name} already exists. "
```

```

        "Check if the target group already exists."
        "Consider using a different name or deleting the existing
target group if appropriate."
    )
    elif error_code == "TooManyTargetGroups":
        log.error(
            "Too many target groups exist in the account. "
            "Consider deleting unused target groups to create space for
new ones."
        )
    log.error(f"Full error:\n\t{err}")

def delete_target_group(self, target_group_name) -> None:
    """
    Deletes the target group.
    """
    try:
        # Describe the target group to get its ARN
        response =
self.elb_client.describe_target_groups(Names=[target_group_name])
        tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

        # Delete the target group
        self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
        log.info("Deleted load balancing target group %s.",
target_group_name)

        # Use a custom waiter to wait until the target group is no longer
available
        self.wait_for_target_group_deletion(self.elb_client, tg_arn)
        log.info("Target group %s successfully deleted.", target_group_name)

    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(f"Failed to delete target group '{target_group_name}'.")
        if error_code == "TargetGroupNotFound":
            log.error(
                "Load balancer target group either already deleted or never
existed. "
                "Verify the name and check that the resource exists in the
AWS Console."
            )
        elif error_code == "ResourceInUseException":

```

```

        log.error(
            "Target group still in use by another resource. "
            "Ensure that the target group is no longer associated with
any load balancers or resources.",
        )
        log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:
                elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn}
still exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts
* delay} seconds."
        )

    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

```

```

:param load_balancer_name: The name of the load balancer to create.
:param subnet_ids: A list of subnets to associate with the load balancer.
:return: Data about the newly created load balancer.
"""
try:
    response = self.elb_client.create_load_balancer(
        Name=load_balancer_name, Subnets=subnet_ids
    )
    load_balancer = response["LoadBalancers"][0]
    log.info(f"Created load balancer '{load_balancer_name}'.")

    waiter = self.elb_client.get_waiter("load_balancer_available")
    log.info(
        f"Waiting for load balancer '{load_balancer_name}' to be
available..."
    )
    waiter.wait(Names=[load_balancer_name])
    log.info(f"Load balancer '{load_balancer_name}' is now available!")

except ClientError as err:
    error_code = err.response["Error"]["Code"]
    log.error(
        f"Failed to create load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
    )

    if error_code == "DuplicateLoadBalancerNameException":
        log.error(
            f"A load balancer with the name '{load_balancer_name}'
already exists. "
            "Load balancer names must be unique within the AWS region. "
            "Please choose a different name and try again."
        )
    if error_code == "TooManyLoadBalancersException":
        log.error(
            "The maximum number of load balancers has been reached in
this account and region. "
            "You can delete unused load balancers or request an increase
in the service quota from AWS Support."
        )
    log.error(f"Full error:\n\t{err}")
else:
    return load_balancer

```



```

def create_listener(
    self,
    load_balancer_name: str,
    target_group: Dict[str, Any],
) -> Dict[str, Any]:
    """
    Creates a listener for the specified load balancer that forwards requests
    to the
    specified target group.

    :param load_balancer_name: The name of the load balancer to create a
    listener for.
    :param target_group: An existing target group that is added as a listener
    to the
                           load balancer.
    :return: Data about the newly created listener.
    """
    try:
        # Retrieve the load balancer ARN
        load_balancer_response = self.elb_client.describe_load_balancers(
            Names=[load_balancer_name]
        )
        load_balancer_arn = load_balancer_response["LoadBalancers"][0][
            "LoadBalancerArn"
        ]

        # Create the listener
        response = self.elb_client.create_listener(
            LoadBalancerArn=load_balancer_arn,
            Protocol=target_group["Protocol"],
            Port=target_group["Port"],
            DefaultActions=[
                {
                    "Type": "forward",
                    "TargetGroupArn": target_group["TargetGroupArn"],
                }
            ],
        )
        log.info(
            f"Created listener to forward traffic from load balancer
            '{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
        )
        return response["Listeners"][0]

```

```

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load
balancer '{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")

def delete_load_balancer(self, load_balancer_name) -> None:
    """
    Deletes a load balancer.

    :param load_balancer_name: The name of the load balancer to delete.
    """
    try:
        response = self.elb_client.describe_load_balancers(
            Names=[load_balancer_name]
        )
        lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
        self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
        log.info("Deleted load balancer %s.", load_balancer_name)
        waiter = self.elb_client.get_waiter("load_balancers_deleted")
        log.info("Waiting for load balancer to be deleted...")
        waiter.wait(Names=[load_balancer_name])
    except ClientError as err:
        error_code = err.response["Error"]["Code"]
        log.error(

```

```

        f"Couldn't delete load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
    )

    if error_code == "LoadBalancerNotFoundException":
        log.error(
            f"The load balancer '{load_balancer_name}' does not exist. "
            "Please check the name and try again."
        )
    log.error(f"Full error:\n\t{err}")

def get_endpoint(self, load_balancer_name) -> str:
    """
    Gets the HTTP endpoint of the load balancer.

    :return: The endpoint.
    """
    try:
        response = self.elb_client.describe_load_balancers(
            Names=[load_balancer_name]
        )
        return response["LoadBalancers"][0]["DNSName"]
    except ClientError as err:
        log.error(
            f"Couldn't get the endpoint for load balancer
{load_balancer_name}"
        )
        error_code = err.response["Error"]["Code"]
        if error_code == "LoadBalancerNotFoundException":
            log.error(
                "Verify load balancer name and ensure it exists in the AWS
console."
            )
        log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.

```

```

        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint,
retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str,
Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not
found. "

```

```

        "Ensure the load balancer exists, is in the correct AWS
region, and "
        "that you have the necessary permissions to access it.",
    )
    elif error_code == "TargetGroupNotFoundException":
        log.error(
            "Target group was not found. "
            "Verify the target group name, check that it exists in the
correct region, "
            "and ensure it has not been deleted or created in a different
account.",
        )
        log.error(f"Full error:\n\t{err}")
    else:
        return health_response["TargetHealthDescriptions"]

```

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: str, dynamodb_client: boto3.client):
        """
        Initializes the RecommendationService class with the necessary
        parameters.

        :param table_name: The name of the DynamoDB recommendations table.
        :param dynamodb_client: A Boto3 DynamoDB client.
        """
        self.table_name = table_name
        self.dynamodb_client = dynamodb_client

    def create(self) -> Dict[str, Any]:
        """

```

```

        Creates a DynamoDB table to use as a recommendation service. The table
        has a
            hash key named 'MediaType' that defines the type of media recommended,
        such as
            Book or Movie, and a range key named 'ItemId' that, combined with the
        MediaType,
            forms a unique identifier for the recommended item.

        :return: Data about the newly created table.
        :raises RecommendationServiceError: If the table creation fails.
        """
        try:
            response = self.dynamodb_client.create_table(
                TableName=self.table_name,
                AttributeDefinitions=[
                    {"AttributeName": "MediaType", "AttributeType": "S"},
                    {"AttributeName": "ItemId", "AttributeType": "N"},
                ],
                KeySchema=[
                    {"AttributeName": "MediaType", "KeyType": "HASH"},
                    {"AttributeName": "ItemId", "KeyType": "RANGE"},
                ],
                ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
            )
            log.info("Creating table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s created.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.

```

```

        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name,
data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}:
{err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.",
self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )

```

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: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
        recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
        These are the name of the DynamoDB recommendation table, no response when
        a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.

```



```

"""
try:
    self.ssm_client.put_parameter(
        Name=name, Value=value, Overwrite=True, Type="String"
    )
    log.info("Setting parameter %s to '%s'.", name, value)
except ClientError as err:
    error_code = err.response["Error"]["Code"]
    log.error(f"Failed to set parameter {name}.")
    if error_code == "ParameterLimitExceeded":
        log.error(
            "The parameter limit has been exceeded. "
            "Consider deleting unused parameters or request a limit
increase."
        )
    elif error_code == "ParameterAlreadyExists":
        log.error(
            "The parameter already exists and overwrite is set to False.
"
            "Use Overwrite=True to update the parameter."
        )
    log.error(f"Full error:\n\t{err}")

```

- 有关 API 详细信息，请参阅《AWS SDK for Python (Boto3) API Reference》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 SDK 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 CLI 创建具有私有子网和 NAT 网关的 VPC。

以下代码示例展示了如何：

- 使用 CLI 创建具有私有子网和 NAT 网关的 VPC。
- 设置必要的组件，包括 VPC、子网、路由表和 NAT 网关。
- 配置安全组和 IAM 角色以实现适当的访问权限和安全性。
- 使用 CLI 命令自动创建和配置这些资源。

Bash

AWS CLI 使用 Bash 脚本

Note

还有更多相关信息 GitHub。在 [Sample developer tutorials](#) 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# VPC with Private Subnets and NAT Gateways (IMDSv2 Compliant Version)
# This script creates a VPC with public and private subnets in two Availability
# Zones,
# NAT gateways, an internet gateway, route tables, a VPC endpoint for S3,
# security groups, a launch template, an Auto Scaling group, and an Application
# Load Balancer.

# Set up logging
LOG_FILE="vpc-private-subnets-nat.log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Cleanup function to delete all created resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete Auto Scaling group if it exists
    if [ -n "${ASG_NAME:-}" ]; then
        echo "Deleting Auto Scaling group: $ASG_NAME"
        aws autoscaling delete-auto-scaling-group --auto-scaling-group-name
"$ASG_NAME" --force-delete
        echo "Waiting for Auto Scaling group to be deleted..."
        aws autoscaling wait auto-scaling-groups-deleted --auto-scaling-group-names
"$ASG_NAME"
    fi

    # Delete load balancer if it exists
    if [ -n "${LB_ARN:-}" ]; then
        echo "Deleting load balancer: $LB_ARN"
        aws elbv2 delete-load-balancer --load-balancer-arn "$LB_ARN"
```

```
# Wait for load balancer to be deleted
sleep 30
fi

# Delete target group if it exists
if [ -n "${TARGET_GROUP_ARN:-}" ]; then
    echo "Deleting target group: $TARGET_GROUP_ARN"
    aws elbv2 delete-target-group --target-group-arn "$TARGET_GROUP_ARN"
fi

# Delete launch template if it exists
if [ -n "${LAUNCH_TEMPLATE_NAME:-}" ]; then
    echo "Deleting launch template: $LAUNCH_TEMPLATE_NAME"
    aws ec2 delete-launch-template --launch-template-name "$LAUNCH_TEMPLATE_NAME"
fi

# Delete NAT Gateways if they exist
if [ -n "${NAT_GW1_ID:-}" ]; then
    echo "Deleting NAT Gateway 1: $NAT_GW1_ID"
    aws ec2 delete-nat-gateway --nat-gateway-id "$NAT_GW1_ID"
fi

if [ -n "${NAT_GW2_ID:-}" ]; then
    echo "Deleting NAT Gateway 2: $NAT_GW2_ID"
    aws ec2 delete-nat-gateway --nat-gateway-id "$NAT_GW2_ID"
fi

# Wait for NAT Gateways to be deleted
if [ -n "${NAT_GW1_ID:-}" ] || [ -n "${NAT_GW2_ID:-}" ]; then
    echo "Waiting for NAT Gateways to be deleted..."
    sleep 60
fi

# Release Elastic IPs if they exist
if [ -n "${EIP1_ALLOC_ID:-}" ]; then
    echo "Releasing Elastic IP 1: $EIP1_ALLOC_ID"
    aws ec2 release-address --allocation-id "$EIP1_ALLOC_ID"
fi

if [ -n "${EIP2_ALLOC_ID:-}" ]; then
    echo "Releasing Elastic IP 2: $EIP2_ALLOC_ID"
    aws ec2 release-address --allocation-id "$EIP2_ALLOC_ID"
fi
```

```
# Delete VPC endpoint if it exists
if [ -n "${VPC_ENDPOINT_ID:-}" ]; then
    echo "Deleting VPC endpoint: $VPC_ENDPOINT_ID"
    aws ec2 delete-vpc-endpoints --vpc-endpoint-ids "$VPC_ENDPOINT_ID"
fi

# Delete security groups if they exist
if [ -n "${APP_SG_ID:-}" ]; then
    echo "Deleting application security group: $APP_SG_ID"
    aws ec2 delete-security-group --group-id "$APP_SG_ID"
fi

if [ -n "${LB_SG_ID:-}" ]; then
    echo "Deleting load balancer security group: $LB_SG_ID"
    aws ec2 delete-security-group --group-id "$LB_SG_ID"
fi

# Detach and delete Internet Gateway if it exists
if [ -n "${IGW_ID:-}" ] && [ -n "${VPC_ID:-}" ]; then
    echo "Detaching Internet Gateway: $IGW_ID from VPC: $VPC_ID"
    aws ec2 detach-internet-gateway --internet-gateway-id "$IGW_ID" --vpc-id
"$VPC_ID"
    echo "Deleting Internet Gateway: $IGW_ID"
    aws ec2 delete-internet-gateway --internet-gateway-id "$IGW_ID"
fi

# Delete route table associations and route tables if they exist
if [ -n "${PUBLIC_RT_ASSOC1_ID:-}" ]; then
    echo "Disassociating public route table from subnet 1: $PUBLIC_RT_ASSOC1_ID"
    aws ec2 disassociate-route-table --association-id "$PUBLIC_RT_ASSOC1_ID"
fi

if [ -n "${PUBLIC_RT_ASSOC2_ID:-}" ]; then
    echo "Disassociating public route table from subnet 2: $PUBLIC_RT_ASSOC2_ID"
    aws ec2 disassociate-route-table --association-id "$PUBLIC_RT_ASSOC2_ID"
fi

if [ -n "${PRIVATE_RT1_ASSOC_ID:-}" ]; then
    echo "Disassociating private route table 1: $PRIVATE_RT1_ASSOC_ID"
    aws ec2 disassociate-route-table --association-id "$PRIVATE_RT1_ASSOC_ID"
fi

if [ -n "${PRIVATE_RT2_ASSOC_ID:-}" ]; then
    echo "Disassociating private route table 2: $PRIVATE_RT2_ASSOC_ID"
```

```
aws ec2 disassociate-route-table --association-id "$PRIVATE_RT2_ASSOC_ID"
fi

if [ -n "${PUBLIC_RT_ID:-}" ]; then
    echo "Deleting public route table: $PUBLIC_RT_ID"
    aws ec2 delete-route-table --route-table-id "$PUBLIC_RT_ID"
fi

if [ -n "${PRIVATE_RT1_ID:-}" ]; then
    echo "Deleting private route table 1: $PRIVATE_RT1_ID"
    aws ec2 delete-route-table --route-table-id "$PRIVATE_RT1_ID"
fi

if [ -n "${PRIVATE_RT2_ID:-}" ]; then
    echo "Deleting private route table 2: $PRIVATE_RT2_ID"
    aws ec2 delete-route-table --route-table-id "$PRIVATE_RT2_ID"
fi

# Delete subnets if they exist
if [ -n "${PUBLIC_SUBNET1_ID:-}" ]; then
    echo "Deleting public subnet 1: $PUBLIC_SUBNET1_ID"
    aws ec2 delete-subnet --subnet-id "$PUBLIC_SUBNET1_ID"
fi

if [ -n "${PUBLIC_SUBNET2_ID:-}" ]; then
    echo "Deleting public subnet 2: $PUBLIC_SUBNET2_ID"
    aws ec2 delete-subnet --subnet-id "$PUBLIC_SUBNET2_ID"
fi

if [ -n "${PRIVATE_SUBNET1_ID:-}" ]; then
    echo "Deleting private subnet 1: $PRIVATE_SUBNET1_ID"
    aws ec2 delete-subnet --subnet-id "$PRIVATE_SUBNET1_ID"
fi

if [ -n "${PRIVATE_SUBNET2_ID:-}" ]; then
    echo "Deleting private subnet 2: $PRIVATE_SUBNET2_ID"
    aws ec2 delete-subnet --subnet-id "$PRIVATE_SUBNET2_ID"
fi

# Delete VPC if it exists
if [ -n "${VPC_ID:-}" ]; then
    echo "Deleting VPC: $VPC_ID"
    aws ec2 delete-vpc --vpc-id "$VPC_ID"
fi
```

```
    echo "Cleanup completed."
}

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command success
check_command() {
    if [ $? -ne 0 ]; then
        handle_error "$1"
    fi
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Create VPC
echo "Creating VPC..."
VPC_RESULT=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
    "ResourceType=vpc,Tags=[{Key=Name,Value=ProductionVPC-$RANDOM_ID}]")
check_command "Failed to create VPC"

VPC_ID=$(echo "$VPC_RESULT" | jq -r '.Vpc.VpcId')
echo "VPC created with ID: $VPC_ID"

# Get Availability Zones
echo "Getting Availability Zones..."
AZ_RESULT=$(aws ec2 describe-availability-zones --query
    'AvailabilityZones[0:2].ZoneName' --output text)
check_command "Failed to get Availability Zones"

# Convert space-separated output to array
read -r -a AZS <<< "$AZ_RESULT"
AZ1=${AZS[0]}
AZ2=${AZS[1]}
echo "Using Availability Zones: $AZ1 and $AZ2"
```

```
# Create subnets
echo "Creating subnets..."
PUBLIC_SUBNET1_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.0.0/24 --availability-zone "$AZ1" --tag-specifications
  "ResourceType=subnet,Tags=[{Key=Name,Value=PublicSubnet1-$RANDOM_ID}]" )
check_command "Failed to create public subnet 1"
PUBLIC_SUBNET1_ID=$(echo "$PUBLIC_SUBNET1_RESULT" | jq -r '.Subnet.SubnetId')

PRIVATE_SUBNET1_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.1.0/24 --availability-zone "$AZ1" --tag-specifications
  "ResourceType=subnet,Tags=[{Key=Name,Value=PrivateSubnet1-$RANDOM_ID}]" )
check_command "Failed to create private subnet 1"
PRIVATE_SUBNET1_ID=$(echo "$PRIVATE_SUBNET1_RESULT" | jq -r '.Subnet.SubnetId')

PUBLIC_SUBNET2_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.2.0/24 --availability-zone "$AZ2" --tag-specifications
  "ResourceType=subnet,Tags=[{Key=Name,Value=PublicSubnet2-$RANDOM_ID}]" )
check_command "Failed to create public subnet 2"
PUBLIC_SUBNET2_ID=$(echo "$PUBLIC_SUBNET2_RESULT" | jq -r '.Subnet.SubnetId')

PRIVATE_SUBNET2_RESULT=$(aws ec2 create-subnet --vpc-id "$VPC_ID" --
cidr-block 10.0.3.0/24 --availability-zone "$AZ2" --tag-specifications
  "ResourceType=subnet,Tags=[{Key=Name,Value=PrivateSubnet2-$RANDOM_ID}]" )
check_command "Failed to create private subnet 2"
PRIVATE_SUBNET2_ID=$(echo "$PRIVATE_SUBNET2_RESULT" | jq -r '.Subnet.SubnetId')

echo "Subnets created with IDs:"
echo "Public Subnet 1: $PUBLIC_SUBNET1_ID"
echo "Private Subnet 1: $PRIVATE_SUBNET1_ID"
echo "Public Subnet 2: $PUBLIC_SUBNET2_ID"
echo "Private Subnet 2: $PRIVATE_SUBNET2_ID"

# Create Internet Gateway
echo "Creating Internet Gateway..."
IGW_RESULT=$(aws ec2 create-internet-gateway --tag-specifications
  "ResourceType=internet-gateway,Tags=[{Key=Name,Value=ProductionIGW-
$RANDOM_ID}]" )
check_command "Failed to create Internet Gateway"
IGW_ID=$(echo "$IGW_RESULT" | jq -r '.InternetGateway.InternetGatewayId')
echo "Internet Gateway created with ID: $IGW_ID"

# Attach Internet Gateway to VPC
echo "Attaching Internet Gateway to VPC..."
```



```
aws ec2 attach-internet-gateway --internet-gateway-id "$IGW_ID" --vpc-id
"$VPC_ID"
check_command "Failed to attach Internet Gateway to VPC"

# Create route tables
echo "Creating route tables..."
PUBLIC_RT_RESULT=$(aws ec2 create-route-table --vpc-id "$VPC_ID" --tag-
specifications "ResourceType=route-table,Tags=[{Key=Name,Value=PublicRouteTable-
$RANDOM_ID}]"")
check_command "Failed to create public route table"
PUBLIC_RT_ID=$(echo "$PUBLIC_RT_RESULT" | jq -r '.RouteTable.RouteTableId')

PRIVATE_RT1_RESULT=$(aws ec2 create-route-table --vpc-
id "$VPC_ID" --tag-specifications "ResourceType=route-
table,Tags=[{Key=Name,Value=PrivateRouteTable1-$RANDOM_ID}]"")
check_command "Failed to create private route table 1"
PRIVATE_RT1_ID=$(echo "$PRIVATE_RT1_RESULT" | jq -r '.RouteTable.RouteTableId')

PRIVATE_RT2_RESULT=$(aws ec2 create-route-table --vpc-
id "$VPC_ID" --tag-specifications "ResourceType=route-
table,Tags=[{Key=Name,Value=PrivateRouteTable2-$RANDOM_ID}]"")
check_command "Failed to create private route table 2"
PRIVATE_RT2_ID=$(echo "$PRIVATE_RT2_RESULT" | jq -r '.RouteTable.RouteTableId')

echo "Route tables created with IDs:"
echo "Public Route Table: $PUBLIC_RT_ID"
echo "Private Route Table 1: $PRIVATE_RT1_ID"
echo "Private Route Table 2: $PRIVATE_RT2_ID"

# Add route to Internet Gateway in public route table
echo "Adding route to Internet Gateway in public route table..."
aws ec2 create-route --route-table-id "$PUBLIC_RT_ID" --destination-cidr-block
0.0.0.0/0 --gateway-id "$IGW_ID"
check_command "Failed to add route to Internet Gateway"

# Associate subnets with route tables
echo "Associating subnets with route tables..."
PUBLIC_RT_ASSOC1_RESULT=$(aws ec2 associate-route-table --route-table-id
"$PUBLIC_RT_ID" --subnet-id "$PUBLIC_SUBNET1_ID")
check_command "Failed to associate public subnet 1 with route table"
PUBLIC_RT_ASSOC1_ID=$(echo "$PUBLIC_RT_ASSOC1_RESULT" | jq -r '.AssociationId')

PUBLIC_RT_ASSOC2_RESULT=$(aws ec2 associate-route-table --route-table-id
"$PUBLIC_RT_ID" --subnet-id "$PUBLIC_SUBNET2_ID")
```

```
check_command "Failed to associate public subnet 2 with route table"
PUBLIC_RT_ASSOC2_ID=$(echo "$PUBLIC_RT_ASSOC2_RESULT" | jq -r '.AssociationId')

PRIVATE_RT1_ASSOC_RESULT=$(aws ec2 associate-route-table --route-table-id
"$PRIVATE_RT1_ID" --subnet-id "$PRIVATE_SUBNET1_ID")
check_command "Failed to associate private subnet 1 with route table"
PRIVATE_RT1_ASSOC_ID=$(echo "$PRIVATE_RT1_ASSOC_RESULT" | jq -r '.AssociationId')

PRIVATE_RT2_ASSOC_RESULT=$(aws ec2 associate-route-table --route-table-id
"$PRIVATE_RT2_ID" --subnet-id "$PRIVATE_SUBNET2_ID")
check_command "Failed to associate private subnet 2 with route table"
PRIVATE_RT2_ASSOC_ID=$(echo "$PRIVATE_RT2_ASSOC_RESULT" | jq -r '.AssociationId')

echo "Route table associations created with IDs:"
echo "Public Subnet 1 Association: $PUBLIC_RT_ASSOC1_ID"
echo "Public Subnet 2 Association: $PUBLIC_RT_ASSOC2_ID"
echo "Private Subnet 1 Association: $PRIVATE_RT1_ASSOC_ID"
echo "Private Subnet 2 Association: $PRIVATE_RT2_ASSOC_ID"

# Create NAT Gateways
echo "Creating NAT Gateways..."

# Allocate Elastic IPs for NAT Gateways
echo "Allocating Elastic IPs for NAT Gateways..."
EIP1_RESULT=$(aws ec2 allocate-address --domain vpc --tag-specifications
"ResourceType=elastic-ip,Tags=[{Key=Name,Value=NAT1-EIP-$RANDOM_ID}]")
check_command "Failed to allocate Elastic IP 1"
EIP1_ALLOC_ID=$(echo "$EIP1_RESULT" | jq -r '.AllocationId')

EIP2_RESULT=$(aws ec2 allocate-address --domain vpc --tag-specifications
"ResourceType=elastic-ip,Tags=[{Key=Name,Value=NAT2-EIP-$RANDOM_ID}]")
check_command "Failed to allocate Elastic IP 2"
EIP2_ALLOC_ID=$(echo "$EIP2_RESULT" | jq -r '.AllocationId')

echo "Elastic IPs allocated with IDs:"
echo "EIP 1 Allocation ID: $EIP1_ALLOC_ID"
echo "EIP 2 Allocation ID: $EIP2_ALLOC_ID"

# Create NAT Gateways
echo "Creating NAT Gateway in public subnet 1..."
NAT_GW1_RESULT=$(aws ec2 create-nat-gateway --subnet-id
"$PUBLIC_SUBNET1_ID" --allocation-id "$EIP1_ALLOC_ID" --tag-specifications
"ResourceType=natgateway,Tags=[{Key=Name,Value=NAT-Gateway1-$RANDOM_ID}]")
check_command "Failed to create NAT Gateway 1"
```

```
NAT_GW1_ID=$(echo "$NAT_GW1_RESULT" | jq -r '.NatGateway.NatGatewayId')

echo "Creating NAT Gateway in public subnet 2..."
NAT_GW2_RESULT=$(aws ec2 create-nat-gateway --subnet-id
"$PUBLIC_SUBNET2_ID" --allocation-id "$EIP2_ALLOC_ID" --tag-specifications
"ResourceType=natgateway,Tags=[{Key=Name,Value=NAT-Gateway2-$RANDOM_ID}]" )
check_command "Failed to create NAT Gateway 2"
NAT_GW2_ID=$(echo "$NAT_GW2_RESULT" | jq -r '.NatGateway.NatGatewayId')

echo "NAT Gateways created with IDs:"
echo "NAT Gateway 1: $NAT_GW1_ID"
echo "NAT Gateway 2: $NAT_GW2_ID"

# Wait for NAT Gateways to be available
echo "Waiting for NAT Gateways to be available..."
aws ec2 wait nat-gateway-available --nat-gateway-ids "$NAT_GW1_ID"
check_command "NAT Gateway 1 did not become available"
aws ec2 wait nat-gateway-available --nat-gateway-ids "$NAT_GW2_ID"
check_command "NAT Gateway 2 did not become available"
echo "NAT Gateways are now available"

# Add routes to NAT Gateways in private route tables
echo "Adding routes to NAT Gateways in private route tables..."
aws ec2 create-route --route-table-id "$PRIVATE_RT1_ID" --destination-cidr-block
0.0.0.0/0 --nat-gateway-id "$NAT_GW1_ID"
check_command "Failed to add route to NAT Gateway 1"

aws ec2 create-route --route-table-id "$PRIVATE_RT2_ID" --destination-cidr-block
0.0.0.0/0 --nat-gateway-id "$NAT_GW2_ID"
check_command "Failed to add route to NAT Gateway 2"

# Create VPC Endpoint for S3
echo "Creating VPC Endpoint for S3..."
S3_PREFIX_LIST_ID=$(aws ec2 describe-prefix-lists --filters "Name=prefix-
list-name,Values=com.amazonaws.$(aws configure get region).s3" --query
'PrefixLists[0].PrefixListId' --output text)
check_command "Failed to get S3 prefix list ID"

VPC_ENDPOINT_RESULT=$(aws ec2 create-vpc-endpoint --vpc-id "$VPC_ID" --
service-name "com.amazonaws.$(aws configure get region).s3" --route-table-ids
"$PRIVATE_RT1_ID" "$PRIVATE_RT2_ID" --tag-specifications "ResourceType=vpc-
endpoint,Tags=[{Key=Name,Value=S3-Endpoint-$RANDOM_ID}]" )
check_command "Failed to create VPC endpoint for S3"
```

```
VPC_ENDPOINT_ID=$(echo "$VPC_ENDPOINT_RESULT" | jq -r
  '.VpcEndpoint.VpcEndpointId')
echo "VPC Endpoint created with ID: $VPC_ENDPOINT_ID"

# Create security groups
echo "Creating security groups..."
LB_SG_RESULT=$(aws ec2 create-security-group --group-name "LoadBalancerSG-
$RANDOM_ID" --description "Security group for the load balancer"
  --vpc-id "$VPC_ID" --tag-specifications "ResourceType=security-
group,Tags=[{Key=Name,Value=LoadBalancerSG-$RANDOM_ID}]" )
check_command "Failed to create load balancer security group"
LB_SG_ID=$(echo "$LB_SG_RESULT" | jq -r '.GroupId')

# Allow inbound HTTP traffic from anywhere to the load balancer
aws ec2 authorize-security-group-ingress --group-id "$LB_SG_ID" --protocol tcp --
port 80 --cidr 0.0.0.0/0
check_command "Failed to authorize ingress to load balancer security group"

APP_SG_RESULT=$(aws ec2 create-security-group --group-name "AppServerSG-
$RANDOM_ID" --description "Security group for the application servers"
  --vpc-id "$VPC_ID" --tag-specifications "ResourceType=security-
group,Tags=[{Key=Name,Value=AppServerSG-$RANDOM_ID}]" )
check_command "Failed to create application server security group"
APP_SG_ID=$(echo "$APP_SG_RESULT" | jq -r '.GroupId')

# Allow inbound HTTP traffic from the load balancer security group to the
  application servers
aws ec2 authorize-security-group-ingress --group-id "$APP_SG_ID" --protocol tcp
  --port 80 --source-group "$LB_SG_ID"
check_command "Failed to authorize ingress to application server security group"

echo "Security groups created with IDs:"
echo "Load Balancer Security Group: $LB_SG_ID"
echo "Application Server Security Group: $APP_SG_ID"

# Create a launch template
echo "Creating launch template..."

# Create user data script with IMDSv2 support
cat > user-data.sh << 'EOF'
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
```

```

systemctl enable httpd

# Use IMDSv2 with session token
TOKEN=$(curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-
metadata-token-ttl-seconds: 21600")
AZ=$(curl -H "X-aws-ec2-metadata-token: $TOKEN" -s http://169.254.169.254/latest/
meta-data/placement/availability-zone)
HOSTNAME=$(hostname -f)

echo "<h1>Hello from $HOSTNAME in $AZ</h1>" > /var/www/html/index.html
EOF

# Encode user data
USER_DATA=$(base64 -w 0 user-data.sh)

# Get latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images --owners amazon --filters
  "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2" "Name=state,Values=available" --
query 'sort_by(Images, &CreationDate)[-1].ImageId' --output text)
check_command "Failed to get latest Amazon Linux 2 AMI"
echo "Using AMI: $AMI_ID"

# Create launch template with IMDSv2 required
LAUNCH_TEMPLATE_NAME="AppServerTemplate-$RANDOM_ID"
echo "Creating launch template: $LAUNCH_TEMPLATE_NAME"

aws ec2 create-launch-template \
  --launch-template-name "$LAUNCH_TEMPLATE_NAME" \
  --version-description "Initial version" \
  --tag-specifications "ResourceType=launch-template,Tags=[{Key=Name,Value=
$LAUNCH_TEMPLATE_NAME}]" \
  --launch-template-data "{
    \"NetworkInterfaces\": [{
      \"DeviceIndex\": 0,
      \"Groups\": [\"$APP_SG_ID\"],
      \"DeleteOnTermination\": true
    }],
    \"ImageId\": \"$AMI_ID\",
    \"InstanceType\": \"t3.micro\",
    \"UserData\": \"$USER_DATA\",
    \"MetadataOptions\": {
      \"HttpTokens\": \"required\",
      \"HttpEndpoint\": \"enabled\"
  }

```

```

    },
    \"TagSpecifications\": [{
      \"ResourceType\": \"instance\",
      \"Tags\": [{
        \"Key\": \"Name\",
        \"Value\": \"AppServer-$RANDOM_ID\"
      }]
    }]
  }
}
}"
check_command "Failed to create launch template"

# Create target group
echo "Creating target group..."
TARGET_GROUP_NAME="AppTargetGroup-$RANDOM_ID"
TARGET_GROUP_RESULT=$(aws elbv2 create-target-group \
  --name "$TARGET_GROUP_NAME" \
  --protocol HTTP \
  --port 80 \
  --vpc-id "$VPC_ID" \
  --target-type instance \
  --health-check-protocol HTTP \
  --health-check-path "/" \
  --health-check-port traffic-port)
check_command "Failed to create target group"
TARGET_GROUP_ARN=$(echo "$TARGET_GROUP_RESULT" | jq -r
  '.TargetGroups[0].TargetGroupArn')
echo "Target group created with ARN: $TARGET_GROUP_ARN"

# Create load balancer
echo "Creating load balancer..."
LB_NAME="AppLoadBalancer-$RANDOM_ID"
LB_RESULT=$(aws elbv2 create-load-balancer \
  --name "$LB_NAME" \
  --subnets "$PUBLIC_SUBNET1_ID" "$PUBLIC_SUBNET2_ID" \
  --security-groups "$LB_SG_ID" \
  --tags "Key=Name,Value=$LB_NAME")
check_command "Failed to create load balancer"
LB_ARN=$(echo "$LB_RESULT" | jq -r '.LoadBalancers[0].LoadBalancerArn')
echo "Load balancer created with ARN: $LB_ARN"

# Wait for load balancer to be active
echo "Waiting for load balancer to be active..."
aws elbv2 wait load-balancer-available --load-balancer-arns "$LB_ARN"
check_command "Load balancer did not become available"

```

```

# Create listener
echo "Creating listener..."
LISTENER_RESULT=$(aws elbv2 create-listener \
  --load-balancer-arn "$LB_ARN" \
  --protocol HTTP \
  --port 80 \
  --default-actions "Type=forward,TargetGroupArn=$TARGET_GROUP_ARN")
check_command "Failed to create listener"
LISTENER_ARN=$(echo "$LISTENER_RESULT" | jq -r '.Listeners[0].ListenerArn')
echo "Listener created with ARN: $LISTENER_ARN"

# Create Auto Scaling group
echo "Creating Auto Scaling group..."
ASG_NAME="AppAutoScalingGroup-$RANDOM_ID"
aws autoscaling create-auto-scaling-group \
  --auto-scaling-group-name "$ASG_NAME" \
  --launch-template "LaunchTemplateName=$LAUNCH_TEMPLATE_NAME,Version=\$Latest" \
  --min-size 2 \
  --max-size 4 \
  --desired-capacity 2 \
  --vpc-zone-identifier "$PRIVATE_SUBNET1_ID,$PRIVATE_SUBNET2_ID" \
  --target-group-arns "$TARGET_GROUP_ARN" \
  --health-check-type ELB \
  --health-check-grace-period 300 \
  --tags "Key=Name,Value=AppServer-$RANDOM_ID,PropagateAtLaunch=true"
check_command "Failed to create Auto Scaling group"
echo "Auto Scaling group created with name: $ASG_NAME"

# Get load balancer DNS name
LB_DNS_NAME=$(aws elbv2 describe-load-balancers --load-balancer-arns "$LB_ARN" --
query 'LoadBalancers[0].DNSName' --output text)
check_command "Failed to get load balancer DNS name"

echo ""
echo "=====
echo "DEPLOYMENT COMPLETE"
echo "=====
echo "VPC ID: $VPC_ID"
echo "Public Subnet 1: $PUBLIC_SUBNET1_ID (AZ: $AZ1)"
echo "Private Subnet 1: $PRIVATE_SUBNET1_ID (AZ: $AZ1)"
echo "Public Subnet 2: $PUBLIC_SUBNET2_ID (AZ: $AZ2)"
echo "Private Subnet 2: $PRIVATE_SUBNET2_ID (AZ: $AZ2)"
echo "NAT Gateway 1: $NAT_GW1_ID"

```

```

echo "NAT Gateway 2: $NAT_GW2_ID"
echo "Load Balancer: $LB_NAME"
echo "Auto Scaling Group: $ASG_NAME"
echo ""
echo "Your application will be available at: http://$LB_DNS_NAME"
echo "It may take a few minutes for the instances to launch and pass health
  checks."
echo ""

# Add health check monitoring
echo "=====
echo "MONITORING INSTANCE HEALTH AND LOAD BALANCER"
echo "=====
echo "Waiting for instances to launch and pass health checks..."
echo "This may take 3-5 minutes. Checking every 30 seconds..."

# Monitor instance health and load balancer accessibility
MAX_ATTEMPTS=10
ATTEMPT=1
HEALTHY_INSTANCES=0

while [ $ATTEMPT -le $MAX_ATTEMPTS ] && [ $HEALTHY_INSTANCES -lt 2 ]; do
  echo "Check attempt $ATTEMPT of $MAX_ATTEMPTS..."

  # Check Auto Scaling group instances
  echo "Checking Auto Scaling group instances..."
  ASG_INSTANCES=$(aws autoscaling describe-auto-scaling-groups --auto-
scaling-group-names "$ASG_NAME" --query 'AutoScalingGroups[0].Instances[*].
[InstanceId,HealthStatus]' --output json)
  echo "ASG Instances status:"
  echo "$ASG_INSTANCES" | jq -r '.[0] | "Instance: \(.Id), Health: \(.HealthStatus)"'

  # Check target group health
  echo "Checking target group health..."
  TARGET_HEALTH=$(aws elbv2 describe-target-health --target-group-arn
"$TARGET_GROUP_ARN" --output json)
  echo "Target health status:"
  echo "$TARGET_HEALTH" | jq -r '.TargetHealthDescriptions[] | "Instance:
\(.TargetId), State: \(.TargetHealth.State), Reason: \(.TargetHealth.Reason //
"N/A"), Description: \(.TargetHealth.Description // "N/A")"'

  # Count healthy instances
  HEALTHY_INSTANCES=$(echo "$TARGET_HEALTH" | jq -r '[_TargetHealthDescriptions[]
| select(.TargetHealth.State=="healthy")] | length')

```



```
echo "Number of healthy instances: $HEALTHY_INSTANCES of 2 expected"

# Check if we have healthy instances
if [ $HEALTHY_INSTANCES -ge 2 ]; then
    echo "All instances are healthy!"

    # Test load balancer accessibility
    echo "Testing load balancer accessibility..."
    HTTP_STATUS=$(curl -s -o /dev/null -w "%{http_code}" "http://$LB_DNS_NAME")

    if [ "$HTTP_STATUS" = "200" ]; then
        echo "Load balancer is accessible! HTTP Status: $HTTP_STATUS"
        echo "You can access your application at: http://$LB_DNS_NAME"

        # Try to get the content to verify IMDSv2 is working
        echo "Fetching content to verify IMDSv2 functionality..."
        CONTENT=$(curl -s "http://$LB_DNS_NAME")
        echo "Response from server:"
        echo "$CONTENT"

        # Check if the content contains the expected pattern
        if [[ "$CONTENT" == *"Hello from"* && "$CONTENT" == *"in"* ]]; then
            echo "IMDSv2 is working correctly! The instance was able to access
metadata using the token-based approach."
        else
            echo "Warning: Content doesn't match expected pattern. IMDSv2
functionality could not be verified."
        fi

        break
    else
        echo "Load balancer returned HTTP status: $HTTP_STATUS"
        echo "Will try again in 30 seconds..."
    fi
else
    echo "Waiting for instances to become healthy..."
    echo "Will check again in 30 seconds..."
fi

ATTEMPT=$((ATTEMPT+1))

if [ $ATTEMPT -le $MAX_ATTEMPTS ]; then
    sleep 30
fi
```

```
done

if [ $HEALTHY_INSTANCES -lt 2 ]; then
    echo "Warning: Not all instances are healthy after maximum attempts."
    echo "You may need to wait longer or check for configuration issues."
fi

echo "To test your application, run:"
echo "curl http://$LB_DNS_NAME"
echo ""
echo "=====
echo "CLEANUP CONFIRMATION"
echo "=====
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been deleted."
else
    echo "Resources will not be deleted. You can manually delete them later."
    echo "To delete resources, run this script again and choose to clean up."
fi
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。

- [AllocateAddress](#)
- [AssociateRouteTable](#)
- [AttachInternetGateway](#)
- [AuthorizeSecurityGroupIngress](#)
- [CreateInternetGateway](#)
- [CreateLaunchTemplate](#)
- [CreateNatGateway](#)
- [CreateRoute](#)
- [CreateRouteTable](#)
- [CreateSecurityGroup](#)
- [CreateSubnet](#)
- [CreateVpc](#)

- [CreateVpcEndpoint](#)
- [DeleteAutoScalingGroup](#)
- [DeleteInternetGateway](#)
- [DeleteLaunchTemplate](#)
- [DeleteLoadBalancer](#)
- [DeleteNatGateway](#)
- [DeleteRouteTable](#)
- [DeleteSecurityGroup](#)
- [DeleteSubnet](#)
- [DeleteTargetGroup](#)
- [DeleteVpc](#)
- [DeleteVpcEndpoints](#)
- [DescribeAvailabilityZones](#)
- [DescribeImages](#)
- [DescribePrefixLists](#)
- [DetachInternetGateway](#)
- [ReleaseAddress](#)

有关 AWS SDK 开发者指南和代码示例的完整列表，请参阅[使用 AWS SDK 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。

开始通过 CLI 使用 Amazon VPC

以下代码示例展示了如何：

- 设置您的 账户
- 创建和配置 VPC
- 配置网络
- 配置安全性
- 部署资源
- 测试和验证
- 清理 资源

- 考虑生产影响
- 考虑安全影响

Bash

AWS CLI 使用 Bash 脚本

Note

还有更多相关信息 GitHub。在 [Sample developer tutorials](#) 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# VPC Creation Script
# This script creates a VPC with public and private subnets, internet gateway,
# NAT gateway, and security groups

# Set up logging
LOG_FILE="vpc_creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"
    do
        echo "- $resource"
    done

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."
```

```

# Reverse the array to delete in reverse order of creation
for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--))
do
    resource="${CREATED_RESOURCES[$i]}"
    resource_type=$(echo "$resource" | cut -d':' -f1)
    resource_id=$(echo "$resource" | cut -d':' -f2)

    case "$resource_type" in
        "INSTANCE")
            echo "Terminating EC2 instance: $resource_id"
            aws ec2 terminate-instances --instance-ids "$resource_id" || echo "Failed
to terminate instance: $resource_id"
            # Wait for instance to terminate
            echo "Waiting for instance to terminate..."
            aws ec2 wait instance-terminated --instance-ids "$resource_id" || echo
"Failed to wait for instance termination: $resource_id"
            ;;
        "KEY_PAIR")
            echo "Deleting key pair: $resource_id"
            aws ec2 delete-key-pair --key-name "$resource_id" || echo "Failed to
delete key pair: $resource_id"
            # Remove the .pem file if it exists
            if [ -f "${resource_id}.pem" ]; then
                rm -f "${resource_id}.pem"
            fi
            ;;
        "NAT_GATEWAY")
            echo "Deleting NAT Gateway: $resource_id"
            aws ec2 delete-nat-gateway --nat-gateway-id "$resource_id" || echo
"Failed to delete NAT Gateway: $resource_id"
            # NAT Gateway deletion takes time, wait for it to complete
            echo "Waiting for NAT Gateway to be deleted..."
            aws ec2 wait nat-gateway-deleted --nat-gateway-ids "$resource_id" || echo
"Failed to wait for NAT Gateway deletion: $resource_id"
            ;;
        "EIP")
            echo "Releasing Elastic IP: $resource_id"
            aws ec2 release-address --allocation-id "$resource_id" || echo "Failed to
release Elastic IP: $resource_id"
            ;;
        "ROUTE_TABLE_ASSOCIATION")
            echo "Disassociating Route Table: $resource_id"

```

```

        aws ec2 disassociate-route-table --association-id "$resource_id" || echo
"Failed to disassociate Route Table: $resource_id"
        ;;
    "ROUTE_TABLE")
        echo "Deleting Route Table: $resource_id"
        aws ec2 delete-route-table --route-table-id "$resource_id" || echo
"Failed to delete Route Table: $resource_id"
        ;;
    "INTERNET_GATEWAY")
        echo "Detaching Internet Gateway: $resource_id from VPC: $VPC_ID"
        aws ec2 detach-internet-gateway --internet-gateway-id "$resource_id" --
vpc-id "$VPC_ID" || echo "Failed to detach Internet Gateway: $resource_id"
        echo "Deleting Internet Gateway: $resource_id"
        aws ec2 delete-internet-gateway --internet-gateway-id "$resource_id" ||
echo "Failed to delete Internet Gateway: $resource_id"
        ;;
    "SECURITY_GROUP")
        echo "Deleting Security Group: $resource_id"
        aws ec2 delete-security-group --group-id "$resource_id" || echo "Failed
to delete Security Group: $resource_id"
        ;;
    "SUBNET")
        echo "Deleting Subnet: $resource_id"
        aws ec2 delete-subnet --subnet-id "$resource_id" || echo "Failed to
delete Subnet: $resource_id"
        ;;
    "VPC")
        echo "Deleting VPC: $resource_id"
        aws ec2 delete-vpc --vpc-id "$resource_id" || echo "Failed to delete VPC:
$resource_id"
        ;;
    esac
done
}

# Initialize array to track created resources
CREATED_RESOURCES=()

echo "Starting VPC creation script at $(date)"

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
aws configure list || handle_error "AWS CLI is not properly configured"

```

```
# Verify identity and permissions
echo "Verifying identity and permissions..."
if ! aws sts get-caller-identity; then
    echo "ERROR: Unable to verify AWS identity. This could be due to:"
    echo "  - Expired credentials"
    echo "  - Missing or invalid AWS credentials"
    echo "  - Insufficient permissions"
    echo ""
    echo "Please run 'aws configure' to update your credentials or check your IAM
permissions."
    exit 1
fi

# Create VPC
echo "Creating VPC with CIDR block 10.0.0.0/16..."
VPC_ID=$(aws ec2 create-vpc --cidr-block 10.0.0.0/16 --tag-specifications
'ResourceType=vpc,Tags=[{Key=Name,Value=MyVPC}]' --query 'Vpc.VpcId' --output
text)

if [ -z "$VPC_ID" ]; then
    handle_error "Failed to create VPC"
fi

CREATED_RESOURCES+=("VPC:$VPC_ID")
echo "VPC created with ID: $VPC_ID"

# Enable DNS support and hostnames
echo "Enabling DNS support and hostnames for VPC..."
aws ec2 modify-vpc-attribute --vpc-id "$VPC_ID" --enable-dns-support ||
    handle_error "Failed to enable DNS support"
aws ec2 modify-vpc-attribute --vpc-id "$VPC_ID" --enable-dns-hostnames ||
    handle_error "Failed to enable DNS hostnames"

# Get available Availability Zones
echo "Getting available Availability Zones..."
AZ1=$(aws ec2 describe-availability-zones --query 'AvailabilityZones[0].ZoneName'
--output text)
AZ2=$(aws ec2 describe-availability-zones --query 'AvailabilityZones[1].ZoneName'
--output text)

if [ -z "$AZ1" ] || [ -z "$AZ2" ]; then
    handle_error "Failed to get Availability Zones"
fi
```

```
echo "Using Availability Zones: $AZ1 and $AZ2"

# Create public subnets
echo "Creating public subnet in $AZ1..."
PUBLIC_SUBNET_AZ1=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
  --cidr-block 10.0.0.0/24 \
  --availability-zone "$AZ1" \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Public-Subnet-
AZ1}]' \
  --query 'Subnet.SubnetId' \
  --output text)

if [ -z "$PUBLIC_SUBNET_AZ1" ]; then
  handle_error "Failed to create public subnet in AZ1"
fi

CREATED_RESOURCES+=("SUBNET:$PUBLIC_SUBNET_AZ1")
echo "Public subnet created in $AZ1 with ID: $PUBLIC_SUBNET_AZ1"

echo "Creating public subnet in $AZ2..."
PUBLIC_SUBNET_AZ2=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
  --cidr-block 10.0.1.0/24 \
  --availability-zone "$AZ2" \
  --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Public-Subnet-
AZ2}]' \
  --query 'Subnet.SubnetId' \
  --output text)

if [ -z "$PUBLIC_SUBNET_AZ2" ]; then
  handle_error "Failed to create public subnet in AZ2"
fi

CREATED_RESOURCES+=("SUBNET:$PUBLIC_SUBNET_AZ2")
echo "Public subnet created in $AZ2 with ID: $PUBLIC_SUBNET_AZ2"

# Create private subnets
echo "Creating private subnet in $AZ1..."
PRIVATE_SUBNET_AZ1=$(aws ec2 create-subnet \
  --vpc-id "$VPC_ID" \
  --cidr-block 10.0.2.0/24 \
  --availability-zone "$AZ1" \
```



```

--tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Private-Subnet-
AZ1}]]' \
--query 'Subnet.SubnetId' \
--output text)

if [ -z "$PRIVATE_SUBNET_AZ1" ]; then
    handle_error "Failed to create private subnet in AZ1"
fi

CREATED_RESOURCES+=("SUBNET:$PRIVATE_SUBNET_AZ1")
echo "Private subnet created in $AZ1 with ID: $PRIVATE_SUBNET_AZ1"

echo "Creating private subnet in $AZ2..."
PRIVATE_SUBNET_AZ2=$(aws ec2 create-subnet \
    --vpc-id "$VPC_ID" \
    --cidr-block 10.0.3.0/24 \
    --availability-zone "$AZ2" \
    --tag-specifications 'ResourceType=subnet,Tags=[{Key=Name,Value=Private-Subnet-
AZ2}]]' \
    --query 'Subnet.SubnetId' \
    --output text)

if [ -z "$PRIVATE_SUBNET_AZ2" ]; then
    handle_error "Failed to create private subnet in AZ2"
fi

CREATED_RESOURCES+=("SUBNET:$PRIVATE_SUBNET_AZ2")
echo "Private subnet created in $AZ2 with ID: $PRIVATE_SUBNET_AZ2"

# Create Internet Gateway
echo "Creating Internet Gateway..."
IGW_ID=$(aws ec2 create-internet-gateway \
    --tag-specifications 'ResourceType=internet-
gateway,Tags=[{Key=Name,Value=MyIGW}]]' \
    --query 'InternetGateway.InternetGatewayId' \
    --output text)

if [ -z "$IGW_ID" ]; then
    handle_error "Failed to create Internet Gateway"
fi

CREATED_RESOURCES+=("INTERNET_GATEWAY:$IGW_ID")
echo "Internet Gateway created with ID: $IGW_ID"

```

```
# Attach Internet Gateway to VPC
echo "Attaching Internet Gateway to VPC..."
aws ec2 attach-internet-gateway --internet-gateway-id "$IGW_ID" --vpc-id
"$VPC_ID" || handle_error "Failed to attach Internet Gateway to VPC"

# Create public route table
echo "Creating public route table..."
PUBLIC_RT=$(aws ec2 create-route-table \
  --vpc-id "$VPC_ID" \
  --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=Public-
RT}]' \
  --query 'RouteTable.RouteTableId' \
  --output text)

if [ -z "$PUBLIC_RT" ]; then
  handle_error "Failed to create public route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE:$PUBLIC_RT")
echo "Public route table created with ID: $PUBLIC_RT"

# Add route to Internet Gateway
echo "Adding route to Internet Gateway in public route table..."
aws ec2 create-route --route-table-id "$PUBLIC_RT" --destination-cidr-block
0.0.0.0/0 --gateway-id "$IGW_ID" || handle_error "Failed to add route to
Internet Gateway"

# Associate public subnets with public route table
echo "Associating public subnet in $AZ1 with public route table..."
PUBLIC_RT_ASSOC_1=$(aws ec2 associate-route-table --route-table-id "$PUBLIC_RT"
--subnet-id "$PUBLIC_SUBNET_AZ1" --query 'AssociationId' --output text)

if [ -z "$PUBLIC_RT_ASSOC_1" ]; then
  handle_error "Failed to associate public subnet in AZ1 with public route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PUBLIC_RT_ASSOC_1")

echo "Associating public subnet in $AZ2 with public route table..."
PUBLIC_RT_ASSOC_2=$(aws ec2 associate-route-table --route-table-id "$PUBLIC_RT"
--subnet-id "$PUBLIC_SUBNET_AZ2" --query 'AssociationId' --output text)

if [ -z "$PUBLIC_RT_ASSOC_2" ]; then
  handle_error "Failed to associate public subnet in AZ2 with public route table"
```

```
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PUBLIC_RT_ASSOC_2")

# Create private route table
echo "Creating private route table..."
PRIVATE_RT=$(aws ec2 create-route-table \
  --vpc-id "$VPC_ID" \
  --tag-specifications 'ResourceType=route-table,Tags=[{Key=Name,Value=Private-RT}]' \
  --query 'RouteTable.RouteTableId' \
  --output text)

if [ -z "$PRIVATE_RT" ]; then
  handle_error "Failed to create private route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE:$PRIVATE_RT")
echo "Private route table created with ID: $PRIVATE_RT"

# Associate private subnets with private route table
echo "Associating private subnet in $AZ1 with private route table..."
PRIVATE_RT_ASSOC_1=$(aws ec2 associate-route-table --route-table-id "$PRIVATE_RT"
  --subnet-id "$PRIVATE_SUBNET_AZ1" --query 'AssociationId' --output text)

if [ -z "$PRIVATE_RT_ASSOC_1" ]; then
  handle_error "Failed to associate private subnet in AZ1 with private route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PRIVATE_RT_ASSOC_1")

echo "Associating private subnet in $AZ2 with private route table..."
PRIVATE_RT_ASSOC_2=$(aws ec2 associate-route-table --route-table-id "$PRIVATE_RT"
  --subnet-id "$PRIVATE_SUBNET_AZ2" --query 'AssociationId' --output text)

if [ -z "$PRIVATE_RT_ASSOC_2" ]; then
  handle_error "Failed to associate private subnet in AZ2 with private route table"
fi

CREATED_RESOURCES+=("ROUTE_TABLE_ASSOCIATION:$PRIVATE_RT_ASSOC_2")

# Allocate Elastic IP for NAT Gateway
```

```
echo "Allocating Elastic IP for NAT Gateway..."
EIP_ALLOC=$(aws ec2 allocate-address --domain vpc --query 'AllocationId' --output
text)

if [ -z "$EIP_ALLOC" ]; then
    handle_error "Failed to allocate Elastic IP"
fi

CREATED_RESOURCES+=("EIP:$EIP_ALLOC")
echo "Elastic IP allocated with ID: $EIP_ALLOC"

# Create NAT Gateway
echo "Creating NAT Gateway in public subnet in $AZ1..."
NAT_GW=$(aws ec2 create-nat-gateway \
    --subnet-id "$PUBLIC_SUBNET_AZ1" \
    --allocation-id "$EIP_ALLOC" \
    --tag-specifications
'ResourceType=natgateway,Tags=[{Key=Name,Value=MyNATGateway}]' \
    --query 'NatGateway.NatGatewayId' \
    --output text)

if [ -z "$NAT_GW" ]; then
    handle_error "Failed to create NAT Gateway"
fi

CREATED_RESOURCES+=("NAT_GATEWAY:$NAT_GW")
echo "NAT Gateway created with ID: $NAT_GW"

# Wait for NAT Gateway to be available
echo "Waiting for NAT Gateway to be available..."
aws ec2 wait nat-gateway-available --nat-gateway-ids "$NAT_GW" || handle_error
"NAT Gateway did not become available"

# Add route to NAT Gateway in private route table
echo "Adding route to NAT Gateway in private route table..."
aws ec2 create-route --route-table-id "$PRIVATE_RT" --destination-cidr-block
0.0.0.0/0 --nat-gateway-id "$NAT_GW" || handle_error "Failed to add route to NAT
Gateway"

# Enable auto-assign public IP for instances in public subnets
echo "Enabling auto-assign public IP for instances in public subnet in $AZ1..."
aws ec2 modify-subnet-attribute --subnet-id "$PUBLIC_SUBNET_AZ1" --map-public-
ip-on-launch || handle_error "Failed to enable auto-assign public IP for public
subnet in AZ1"
```

```
echo "Enabling auto-assign public IP for instances in public subnet in $AZ2..."
aws ec2 modify-subnet-attribute --subnet-id "$PUBLIC_SUBNET_AZ2" --map-public-
ip-on-launch || handle_error "Failed to enable auto-assign public IP for public
  subnet in AZ2"

# Create security group for web servers
echo "Creating security group for web servers..."
WEB_SG=$(aws ec2 create-security-group \
  --group-name "WebServerSG-$(date +%s)" \
  --description "Security group for web servers" \
  --vpc-id "$VPC_ID" \
  --query 'GroupId' \
  --output text)

if [ -z "$WEB_SG" ]; then
  handle_error "Failed to create security group for web servers"
fi

CREATED_RESOURCES+=("SECURITY_GROUP:$WEB_SG")
echo "Security group for web servers created with ID: $WEB_SG"

# Allow HTTP and HTTPS traffic
echo "Allowing HTTP traffic to web servers security group..."
aws ec2 authorize-security-group-ingress --group-id "$WEB_SG" --protocol tcp --
port 80 --cidr 0.0.0.0/0 || handle_error "Failed to allow HTTP traffic"

echo "Allowing HTTPS traffic to web servers security group..."
aws ec2 authorize-security-group-ingress --group-id "$WEB_SG" --protocol tcp --
port 443 --cidr 0.0.0.0/0 || handle_error "Failed to allow HTTPS traffic"

# Note: In a production environment, you should restrict the source IP ranges for
  security
echo "NOTE: In a production environment, you should restrict the source IP ranges
  for HTTP and HTTPS traffic"

# Create security group for database servers
echo "Creating security group for database servers..."
DB_SG=$(aws ec2 create-security-group \
  --group-name "DBServerSG-$(date +%s)" \
  --description "Security group for database servers" \
  --vpc-id "$VPC_ID" \
  --query 'GroupId' \
  --output text)
```

```
if [ -z "$DB_SG" ]; then
    handle_error "Failed to create security group for database servers"
fi

CREATED_RESOURCES+=("SECURITY_GROUP:$DB_SG")
echo "Security group for database servers created with ID: $DB_SG"

# Allow MySQL/Aurora traffic from web servers only
echo "Allowing MySQL/Aurora traffic from web servers to database servers..."
aws ec2 authorize-security-group-ingress --group-id "$DB_SG" --protocol tcp --
port 3306 --source-group "$WEB_SG" || handle_error "Failed to allow MySQL/Aurora
traffic"

# Verify VPC configuration
echo "Verifying VPC configuration..."
echo "VPC:"
aws ec2 describe-vpcs --vpc-id "$VPC_ID" || handle_error "Failed to describe VPC"

echo "Subnets:"
aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID" || handle_error
"Failed to describe subnets"

echo "Route tables:"
aws ec2 describe-route-tables --filters "Name=vpc-id,Values=$VPC_ID" ||
handle_error "Failed to describe route tables"

echo "Internet gateway:"
aws ec2 describe-internet-gateways --filters "Name=attachment.vpc-id,Values=
$VPC_ID" || handle_error "Failed to describe Internet Gateway"

echo "NAT gateway:"
aws ec2 describe-nat-gateways --filter "Name=vpc-id,Values=$VPC_ID" ||
handle_error "Failed to describe NAT Gateway"

echo "Security groups:"
aws ec2 describe-security-groups --filters "Name=vpc-id,Values=$VPC_ID" ||
handle_error "Failed to describe security groups"

echo ""
# Summary of created resources
echo "VPC creation completed successfully!"
echo "Summary of created resources:"
echo "- VPC: $VPC_ID"
```

```

echo "- Public Subnet in $AZ1: $PUBLIC_SUBNET_AZ1"
echo "- Public Subnet in $AZ2: $PUBLIC_SUBNET_AZ2"
echo "- Private Subnet in $AZ1: $PRIVATE_SUBNET_AZ1"
echo "- Private Subnet in $AZ2: $PRIVATE_SUBNET_AZ2"
echo "- Internet Gateway: $IGW_ID"
echo "- Public Route Table: $PUBLIC_RT"
echo "- Private Route Table: $PRIVATE_RT"
echo "- Elastic IP: $EIP_ALLOC"
echo "- NAT Gateway: $NAT_GW"
echo "- Web Servers Security Group: $WEB_SG"
echo "- Database Servers Security Group: $DB_SG"

# Deploy EC2 instances
echo ""
echo "Deploying EC2 instances..."

# Create key pair for SSH access
KEY_NAME="vpc-tutorial-key-$(date +%s)"
echo "Creating key pair $KEY_NAME..."
aws ec2 create-key-pair --key-name "$KEY_NAME" --query 'KeyMaterial' --output
  text > "${KEY_NAME}.pem" || handle_error "Failed to create key pair"
chmod 400 "${KEY_NAME}.pem"
echo "Key pair saved to ${KEY_NAME}.pem"
CREATED_RESOURCES+=("KEY_PAIR:$KEY_NAME")

# Get latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*x86_64-gp2"
  "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" --output text) ||
  handle_error "Failed to get AMI"
echo "Using AMI: $AMI_ID"

# Launch web server in public subnet
echo "Launching web server in public subnet..."
WEB_INSTANCE=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --count 1 \
  --instance-type t2.micro \
  --key-name "$KEY_NAME" \
  --security-group-ids "$WEB_SG" \
  --subnet-id "$PUBLIC_SUBNET_AZ1" \
  --associate-public-ip-address \

```

```

--user-data '#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
systemctl enable httpd
echo "<h1>Hello from $(hostname -f) in the public subnet</h1>" > /var/www/
html/index.html' \
--tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=WebServer}]' \
\
--query 'Instances[0].InstanceId' \
--output text) || handle_error "Failed to launch web server"
echo "Web server instance created with ID: $WEB_INSTANCE"
CREATED_RESOURCES+=("INSTANCE:$WEB_INSTANCE")

# Wait for web server to be running
echo "Waiting for web server to be running..."
aws ec2 wait instance-running --instance-ids "$WEB_INSTANCE"

# Get web server public IP
WEB_PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$WEB_INSTANCE" \
--query 'Reservations[0].Instances[0].PublicIpAddress' --output text)
echo "Web server public IP: $WEB_PUBLIC_IP"
echo "You can access the web server at: http://$WEB_PUBLIC_IP"

# Launch database server in private subnet
echo "Launching database server in private subnet..."
DB_INSTANCE=$(aws ec2 run-instances \
--image-id "$AMI_ID" \
--count 1 \
--instance-type t2.micro \
--key-name "$KEY_NAME" \
--security-group-ids "$DB_SG" \
--subnet-id "$PRIVATE_SUBNET_AZ1" \
--user-data '#!/bin/bash
yum update -y
yum install -y mariadb-server
systemctl start mariadb
systemctl enable mariadb' \
--tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=DBServer}]' \
--query 'Instances[0].InstanceId' \
--output text) || handle_error "Failed to launch database server"
echo "Database server instance created with ID: $DB_INSTANCE"
CREATED_RESOURCES+=("INSTANCE:$DB_INSTANCE")

```



```
# Wait for database server to be running
echo "Waiting for database server to be running..."
aws ec2 wait instance-running --instance-ids "$DB_INSTANCE"

# Get database server private IP
DB_PRIVATE_IP=$(aws ec2 describe-instances --instance-ids "$DB_INSTANCE" \
  --query 'Reservations[0].Instances[0].PrivateIpAddress' --output text)
echo "Database server private IP: $DB_PRIVATE_IP"

echo "EC2 instances deployed successfully!"
echo "- Web Server (Public): $WEB_INSTANCE ($WEB_PUBLIC_IP)"
echo "- Database Server (Private): $DB_INSTANCE ($DB_PRIVATE_IP)"
echo ""
echo "Note: To connect to the web server: ssh -i ${KEY_NAME}.pem ec2-user@
$WEB_PUBLIC_IP"
echo "To connect to the database server, you must first connect to the web
server, then use it as a bastion host."
echo "======"
echo "CLEANUP CONFIRMATION"
echo "======"
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE
if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
  echo "Cleaning up resources..."
  cleanup_resources
  echo "All resources have been cleaned up."
else
  echo "Resources will not be cleaned up. You can manually clean them up later."
fi

echo "Script completed at $(date)"
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。

- [AllocateAddress](#)
- [AssociateRouteTable](#)
- [AttachInternetGateway](#)
- [AuthorizeSecurityGroupIngress](#)
- [CreateInternetGateway](#)
- [CreateKeyPair](#)

- [CreateNatGateway](#)
- [CreateRoute](#)
- [CreateRouteTable](#)
- [CreateSecurityGroup](#)
- [CreateSubnet](#)
- [CreateVpc](#)
- [DeleteInternetGateway](#)
- [DeleteKeyPair](#)
- [DeleteNatGateway](#)
- [DeleteRouteTable](#)
- [DeleteSecurityGroup](#)
- [DeleteSubnet](#)
- [DeleteVpc](#)
- [DescribeAvailabilityZones](#)
- [DescribeImages](#)
- [DescribeInstances](#)
- [DescribeInternetGateways](#)
- [DescribeNatGateways](#)
- [DescribeRouteTables](#)
- [DescribeSecurityGroups](#)
- [DescribeSubnets](#)
- [DescribeVpcs](#)
- [DetachInternetGateway](#)
- [DisassociateRouteTable](#)
- [ModifySubnetAttribute](#)
- [ModifyVpcAttribute](#)
- [ReleaseAddress](#)
- [RunInstances](#)
- [TerminateInstances](#)

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 SDK 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 CLI 的中转网关入门

以下代码示例展示了如何：

- 创建具有 DNS 支持和默认路由表设置的中转网关
- 等待中转网关变为可用
- 使用子网将两个 VPCs 连接到传输网关
- 等待 VPC 附件变为可用
- VPCs 通过公交网关添加往返路线
- 测试 VPC 间的连接
- 清理资源，包括路由、附件和中转网关

Bash

AWS CLI 使用 Bash 脚本

Note

还有更多相关信息 GitHub。在 [Sample developer tutorials](#) 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Amazon VPC Transit Gateway CLI Script
# This script demonstrates how to create a transit gateway and connect two VPCs
# Modified to work with older AWS CLI versions that don't support transit gateway
# wait commands

# Error handling
set -e
LOG_FILE="transit-gateway-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

# Function to wait for transit gateway to be available
wait_for_tgw() {
```

```
local tgw_id=$1
echo "Waiting for Transit Gateway $tgw_id to become available..."

while true; do
    status=$(aws ec2 describe-transit-gateways --transit-gateway-ids "$tgw_id" --
query "TransitGateways[0].State" --output text)
    echo "Current status: $status"

    if [ "$status" = "available" ]; then
        echo "Transit Gateway is now available"
        break
    fi

    echo "Waiting for transit gateway to become available. Current state:
$status"
    sleep 10
done
}

# Function to wait for transit gateway attachment to be available
wait_for_tgw_attachment() {
    local attachment_id=$1
    echo "Waiting for Transit Gateway Attachment $attachment_id to become
available..."

    while true; do
        status=$(aws ec2 describe-transit-gateway-vpc-attachments --transit-gateway-
attachment-ids "$attachment_id" --query "TransitGatewayVpcAttachments[0].State"
--output text)
        echo "Current status: $status"

        if [ "$status" = "available" ]; then
            echo "Transit Gateway Attachment is now available"
            break
        fi

        echo "Waiting for transit gateway attachment to become available. Current
state: $status"
        sleep 10
    done
}

# Function to wait for transit gateway attachment to be deleted
wait_for_tgw_attachment_deleted() {
```

```
local attachment_id=$1
echo "Waiting for Transit Gateway Attachment $attachment_id to be deleted..."

while true; do
    # Check if the attachment still exists
    count=$(aws ec2 describe-transit-gateway-vpc-attachments --filters
"Name=transit-gateway-attachment-id,Values=$attachment_id" --query
"length(TransitGatewayVpcAttachments)" --output text)

    if [ "$count" = "0" ]; then
        echo "Transit Gateway Attachment has been deleted"
        break
    fi

    status=$(aws ec2 describe-transit-gateway-vpc-attachments --transit-gateway-
attachment-ids "$attachment_id" --query "TransitGatewayVpcAttachments[0].State"
--output text 2>/dev/null || echo "deleted")

    if [ "$status" = "deleted" ]; then
        echo "Transit Gateway Attachment has been deleted"
        break
    fi

    echo "Waiting for transit gateway attachment to be deleted. Current state:
$status"
    sleep 10
done
}

# Function to clean up resources
cleanup() {
    echo "Error occurred. Cleaning up resources..."

    # Delete resources in reverse order
    if [ ! -z "$TGW_ATTACHMENT_1_ID" ]; then
        echo "Deleting Transit Gateway VPC Attachment 1: $TGW_ATTACHMENT_1_ID"
        aws ec2 delete-transit-gateway-vpc-attachment --transit-gateway-attachment-id
"$TGW_ATTACHMENT_1_ID" || true
        wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_1_ID" || true
    fi

    if [ ! -z "$TGW_ATTACHMENT_2_ID" ]; then
        echo "Deleting Transit Gateway VPC Attachment 2: $TGW_ATTACHMENT_2_ID"
```

```
aws ec2 delete-transit-gateway-vpc-attachment --transit-gateway-attachment-id
"$TGW_ATTACHMENT_2_ID" || true
wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_2_ID" || true
fi

if [ ! -z "$TGW_ID" ]; then
    echo "Deleting Transit Gateway: $TGW_ID"
    aws ec2 delete-transit-gateway --transit-gateway-id "$TGW_ID" || true
fi

exit 1
}

# Set up trap for error handling
trap cleanup ERR

echo "=== Amazon VPC Transit Gateway Tutorial ==="
echo "This script will create a transit gateway and connect two VPCs"
echo ""

# Get a valid availability zone dynamically
echo "Getting available AZ in current region..."
AZ=$(aws ec2 describe-availability-zones --query "AvailabilityZones[0].ZoneName"
--output text)
echo "Using availability zone: $AZ"

# Check if VPCs exist
echo "Checking for existing VPCs..."
VPC1_ID=$(aws ec2 describe-vpcs --filters "Name=tag:Name,Values=VPC1" --query
"Vpcs[0].VpcId" --output text)
VPC2_ID=$(aws ec2 describe-vpcs --filters "Name=tag:Name,Values=VPC2" --query
"Vpcs[0].VpcId" --output text)

if [ "$VPC1_ID" == "None" ] || [ -z "$VPC1_ID" ]; then
    echo "Creating VPC1..."
    VPC1_ID=$(aws ec2 create-vpc --cidr-block 10.1.0.0/16 --tag-specifications
'ResourceType=vpc,Tags=[{Key=Name,Value=VPC1}]' --query Vpc.VpcId --output text)
    echo "Created VPC1: $VPC1_ID"

    # Create a subnet in VPC1
    echo "Creating subnet in VPC1..."
    SUBNET1_ID=$(aws ec2 create-subnet --vpc-id "$VPC1_ID" --cidr-
block 10.1.0.0/24 --availability-zone "$AZ" --tag-specifications
```

```

'ResourceType=subnet,Tags=[{Key=Name,Value=VPC1-Subnet}]' --query
Subnet.SubnetId --output text)
    echo "Created subnet in VPC1: $SUBNET1_ID"
else
    echo "Using existing VPC1: $VPC1_ID"
    SUBNET1_ID=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC1_ID"
--query "Subnets[0].SubnetId" --output text)
    if [ "$SUBNET1_ID" == "None" ] || [ -z "$SUBNET1_ID" ]; then
        echo "Creating subnet in VPC1..."
        SUBNET1_ID=$(aws ec2 create-subnet --vpc-id "$VPC1_ID" --cidr-
block 10.1.0.0/24 --availability-zone "$AZ" --tag-specifications
'ResourceType=subnet,Tags=[{Key=Name,Value=VPC1-Subnet}]' --query
Subnet.SubnetId --output text)
        echo "Created subnet in VPC1: $SUBNET1_ID"
    else
        echo "Using existing subnet in VPC1: $SUBNET1_ID"
    fi
fi

if [ "$VPC2_ID" == "None" ] || [ -z "$VPC2_ID" ]; then
    echo "Creating VPC2..."
    VPC2_ID=$(aws ec2 create-vpc --cidr-block 10.2.0.0/16 --tag-specifications
'ResourceType=vpc,Tags=[{Key=Name,Value=VPC2}]' --query Vpc.VpcId --output text)
    echo "Created VPC2: $VPC2_ID"

    # Create a subnet in VPC2
    echo "Creating subnet in VPC2..."
    SUBNET2_ID=$(aws ec2 create-subnet --vpc-id "$VPC2_ID" --cidr-
block 10.2.0.0/24 --availability-zone "$AZ" --tag-specifications
'ResourceType=subnet,Tags=[{Key=Name,Value=VPC2-Subnet}]' --query
Subnet.SubnetId --output text)
    echo "Created subnet in VPC2: $SUBNET2_ID"
else
    echo "Using existing VPC2: $VPC2_ID"
    SUBNET2_ID=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC2_ID"
--query "Subnets[0].SubnetId" --output text)
    if [ "$SUBNET2_ID" == "None" ] || [ -z "$SUBNET2_ID" ]; then
        echo "Creating subnet in VPC2..."
        SUBNET2_ID=$(aws ec2 create-subnet --vpc-id "$VPC2_ID" --cidr-
block 10.2.0.0/24 --availability-zone "$AZ" --tag-specifications
'ResourceType=subnet,Tags=[{Key=Name,Value=VPC2-Subnet}]' --query
Subnet.SubnetId --output text)
        echo "Created subnet in VPC2: $SUBNET2_ID"
    else

```

```

    echo "Using existing subnet in VPC2: $SUBNET2_ID"
  fi
fi

# Get route tables for each VPC
RTB1_ID=$(aws ec2 describe-route-tables --filters "Name=vpc-id,Values=$VPC1_ID"
--query "RouteTables[0].RouteTableId" --output text)
RTB2_ID=$(aws ec2 describe-route-tables --filters "Name=vpc-id,Values=$VPC2_ID"
--query "RouteTables[0].RouteTableId" --output text)

echo "Route table for VPC1: $RTB1_ID"
echo "Route table for VPC2: $RTB2_ID"

# Step 1: Create the transit gateway
echo "Creating Transit Gateway..."
TGW_ID=$(aws ec2 create-transit-gateway \
  --description "My Transit Gateway" \
  --options
AmazonSideAsn=64512,AutoAcceptSharedAttachments=disable,DefaultRouteTableAssociation=enable \
  --tag-specifications 'ResourceType=transit-
gateway,Tags=[{Key=Name,Value=MyTransitGateway}]' \
  --query TransitGateway.TransitGatewayId \
  --output text)

echo "Created Transit Gateway: $TGW_ID"

# Wait for the transit gateway to become available
wait_for_tgw "$TGW_ID"

# Step 2: Attach VPCs to the transit gateway
echo "Attaching VPC1 to Transit Gateway..."
TGW_ATTACHMENT_1_ID=$(aws ec2 create-transit-gateway-vpc-attachment \
  --transit-gateway-id "$TGW_ID" \
  --vpc-id "$VPC1_ID" \
  --subnet-ids "$SUBNET1_ID" \
  --tag-specifications 'ResourceType=transit-gateway-
attachment,Tags=[{Key=Name,Value=VPC1-Attachment}]' \
  --query TransitGatewayVpcAttachment.TransitGatewayAttachmentId \
  --output text)

echo "Created Transit Gateway VPC Attachment for VPC1: $TGW_ATTACHMENT_1_ID"

echo "Attaching VPC2 to Transit Gateway..."

```



```
TGW_ATTACHMENT_2_ID=$(aws ec2 create-transit-gateway-vpc-attachment \
  --transit-gateway-id "$TGW_ID" \
  --vpc-id "$VPC2_ID" \
  --subnet-ids "$SUBNET2_ID" \
  --tag-specifications 'ResourceType=transit-gateway-
attachment,Tags=[{Key=Name,Value=VPC2-Attachment}]' \
  --query TransitGatewayVpcAttachment.TransitGatewayAttachmentId \
  --output text)

echo "Created Transit Gateway VPC Attachment for VPC2: $TGW_ATTACHMENT_2_ID"

# Wait for the attachments to become available
wait_for_tgw_attachment "$TGW_ATTACHMENT_1_ID"
wait_for_tgw_attachment "$TGW_ATTACHMENT_2_ID"

# Step 3: Add routes between the transit gateway and VPCs
echo "Adding route from VPC1 to VPC2 via Transit Gateway..."
aws ec2 create-route \
  --route-table-id "$RTB1_ID" \
  --destination-cidr-block 10.2.0.0/16 \
  --transit-gateway-id "$TGW_ID"

echo "Adding route from VPC2 to VPC1 via Transit Gateway..."
aws ec2 create-route \
  --route-table-id "$RTB2_ID" \
  --destination-cidr-block 10.1.0.0/16 \
  --transit-gateway-id "$TGW_ID"

echo "Routes added successfully"

# Step 4: Display information for testing
echo ""
echo "=== Transit Gateway Setup Complete ==="
echo "Transit Gateway ID: $TGW_ID"
echo "VPC1 ID: $VPC1_ID"
echo "VPC2 ID: $VPC2_ID"
echo ""
echo "To test connectivity:"
echo "1. Launch an EC2 instance in each VPC"
echo "2. Configure security groups to allow ICMP traffic"
echo "3. Connect to one instance and ping the other instance's private IP"
echo ""

# Prompt user before cleanup
```

```
read -p "Press Enter to view created resources, or Ctrl+C to exit without
cleanup..."

echo ""
echo "=== Resources Created ==="
echo "Transit Gateway: $TGW_ID"
echo "VPC1: $VPC1_ID"
echo "VPC2: $VPC2_ID"
echo "Subnet in VPC1: $SUBNET1_ID"
echo "Subnet in VPC2: $SUBNET2_ID"
echo "Transit Gateway Attachment for VPC1: $TGW_ATTACHMENT_1_ID"
echo "Transit Gateway Attachment for VPC2: $TGW_ATTACHMENT_2_ID"
echo ""

read -p "Do you want to clean up these resources? (y/n): " CLEANUP_CONFIRM
if [[ $CLEANUP_CONFIRM == "y" || $CLEANUP_CONFIRM == "Y" ]]; then
    echo "Starting cleanup..."

    # Delete routes
    echo "Deleting routes..."
    aws ec2 delete-route --route-table-id "$RTB1_ID" --destination-cidr-block
10.2.0.0/16
    aws ec2 delete-route --route-table-id "$RTB2_ID" --destination-cidr-block
10.1.0.0/16

    # Delete transit gateway attachments
    echo "Deleting Transit Gateway VPC Attachment for VPC1: $TGW_ATTACHMENT_1_ID"
    aws ec2 delete-transit-gateway-vpc-attachment --transit-gateway-attachment-id
"$TGW_ATTACHMENT_1_ID"

    echo "Deleting Transit Gateway VPC Attachment for VPC2: $TGW_ATTACHMENT_2_ID"
    aws ec2 delete-transit-gateway-vpc-attachment --transit-gateway-attachment-id
"$TGW_ATTACHMENT_2_ID"

    # Wait for attachments to be deleted
    wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_1_ID"
    wait_for_tgw_attachment_deleted "$TGW_ATTACHMENT_2_ID"

    # Delete transit gateway
    echo "Deleting Transit Gateway: $TGW_ID"
    aws ec2 delete-transit-gateway --transit-gateway-id "$TGW_ID"

    echo "Cleanup completed successfully"
else
```

```
    echo "Skipping cleanup. Resources will continue to incur charges until manually
    deleted."
fi

echo "Tutorial completed. See $LOG_FILE for detailed logs."
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。

- [CreateRoute](#)
- [CreateSubnet](#)
- [CreateTransitGateway](#)
- [CreateTransitGatewayVpcAttachment](#)
- [CreateVpc](#)
- [DeleteRoute](#)
- [DeleteTransitGateway](#)
- [DeleteTransitGatewayVpcAttachment](#)
- [DescribeAvailabilityZones](#)
- [DescribeRouteTables](#)
- [DescribeSubnets](#)
- [DescribeTransitGatewayAttachments](#)
- [DescribeTransitGatewayVpcAttachments](#)
- [DescribeTransitGateways](#)
- [DescribeVpcs](#)

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 SDK 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。

开始通过 CLI 使用 Amazon VPC IPAM

以下代码示例展示了如何：

- 使用 CLI 设置和配置 Amazon VPC IP 地址管理器 (IPAM)。
- 创建包含运营区域 (例如 us-east-1、us-west-2) 的 IPAM。
- 检索 IPAM 的私有范围 ID。
- 创建 IPv4 池的分层结构 (顶级池、区域池和开发池)。

- 为每个池预置 CIDR 块 (例如 10.0.0.0/8、10.0.0.0/16、10.0.0.0/24) 。
- 使用从 IPAM 池分配的 CIDR 创建 VPC。
- 验证 IPAM 池分配和 VPC 创建情况。
- 对权限错误、CIDR 分配失败和依赖项冲突等常见问题进行故障排除。
- 清理 IPAM 资源 (VPC、池和 IPAM) CIDRs , 以避免不必要的费用。
- 探索 IPAM 高级 IPAM 功能的后续步骤。

Bash

AWS CLI 使用 Bash 脚本

Note

还有更多相关信息 GitHub。在 [Sample developer tutorials](#) 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# IPAM Getting Started CLI Script - Version 7
# This script creates an IPAM, creates a hierarchy of IP address pools, and
# allocates a CIDR to a VPC
# Fixed to correctly identify the private scope ID, wait for resources to be
# available, add locale to development pool,
# use the correct parameter names for VPC creation, and wait for CIDR
# provisioning to complete

# Set up logging
LOG_FILE="ipam_script.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting IPAM setup script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
```

```

    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo ""
    echo "=====
    echo "RESOURCES CREATED:"
    echo "=====

    if [ -n "$VPC_ID" ]; then
        echo "VPC: $VPC_ID"
    fi

    if [ -n "$DEV_POOL_ID" ]; then
        echo "Development Pool: $DEV_POOL_ID"
    fi

    if [ -n "$REGIONAL_POOL_ID" ]; then
        echo "Regional Pool: $REGIONAL_POOL_ID"
    fi

    if [ -n "$TOP_POOL_ID" ]; then
        echo "Top-level Pool: $TOP_POOL_ID"
    fi

    if [ -n "$IPAM_ID" ]; then
        echo "IPAM: $IPAM_ID"
    fi

    echo ""
    echo "=====
    echo "CLEANUP CONFIRMATION"
    echo "=====
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."

        # Delete resources in reverse order of creation to handle dependencies

        if [ -n "$VPC_ID" ]; then

```

```

        echo "Deleting VPC: $VPC_ID"
        aws ec2 delete-vpc --vpc-id "$VPC_ID" || echo "Failed to delete VPC"
        echo "Waiting for VPC to be deleted..."
        sleep 10
    fi

    if [ -n "$DEV_POOL_ID" ]; then
        echo "Deleting Development Pool: $DEV_POOL_ID"
        # First deprovision any CIDRs from the pool
        CIDRS=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id "$DEV_POOL_ID" --
query 'IpamPoolCidrs[].Cidr' --output text)
        for CIDR in $CIDRS; do
            echo "Deprovisioning CIDR $CIDR from Development Pool"
            aws ec2 deprovision-ipam-pool-cidr --ipam-pool-id "$DEV_POOL_ID"
--cidr "$CIDR" || echo "Failed to deprovision CIDR $CIDR"
            sleep 5
        done
        aws ec2 delete-ipam-pool --ipam-pool-id "$DEV_POOL_ID" || echo
"Failed to delete Development Pool"
        echo "Waiting for Development Pool to be deleted..."
        sleep 10
    fi

    if [ -n "$REGIONAL_POOL_ID" ]; then
        echo "Deleting Regional Pool: $REGIONAL_POOL_ID"
        # First deprovision any CIDRs from the pool
        CIDRS=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id
"$REGIONAL_POOL_ID" --query 'IpamPoolCidrs[].Cidr' --output text)
        for CIDR in $CIDRS; do
            echo "Deprovisioning CIDR $CIDR from Regional Pool"
            aws ec2 deprovision-ipam-pool-cidr --ipam-pool-id
"$REGIONAL_POOL_ID" --cidr "$CIDR" || echo "Failed to deprovision CIDR $CIDR"
            sleep 5
        done
        aws ec2 delete-ipam-pool --ipam-pool-id "$REGIONAL_POOL_ID" || echo
"Failed to delete Regional Pool"
        echo "Waiting for Regional Pool to be deleted..."
        sleep 10
    fi

    if [ -n "$TOP_POOL_ID" ]; then
        echo "Deleting Top-level Pool: $TOP_POOL_ID"
        # First deprovision any CIDRs from the pool

```

```

        CIDRS=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id "$TOP_POOL_ID" --
query 'IpamPoolCidrs[].Cidr' --output text)
        for CIDR in $CIDRS; do
            echo "Deprovisioning CIDR $CIDR from Top-level Pool"
            aws ec2 deprovision-ipam-pool-cidr --ipam-pool-id "$TOP_POOL_ID"
--cidr "$CIDR" || echo "Failed to deprovision CIDR $CIDR"
            sleep 5
        done
        aws ec2 delete-ipam-pool --ipam-pool-id "$TOP_POOL_ID" || echo
"Failed to delete Top-level Pool"
        echo "Waiting for Top-level Pool to be deleted..."
        sleep 10
    fi

    if [ -n "$IPAM_ID" ]; then
        echo "Deleting IPAM: $IPAM_ID"
        aws ec2 delete-ipam --ipam-id "$IPAM_ID" || echo "Failed to delete
IPAM"
    fi

    echo "Cleanup completed."
else
    echo "Cleanup skipped. Resources will remain in your account."
fi
}

# Function to wait for a pool to be in the 'create-complete' state
wait_for_pool() {
    local pool_id=$1
    local max_attempts=30
    local attempt=1
    local state=""

    echo "Waiting for pool $pool_id to be available..."

    while [ $attempt -le $max_attempts ]; do
        state=$(aws ec2 describe-ipam-pools --ipam-pool-ids "$pool_id" --query
'IpamPools[0].State' --output text)

        if [ "$state" = "create-complete" ]; then
            echo "Pool $pool_id is now available (state: $state)"
            return 0
        fi
    done
}

```

```

        echo "Attempt $attempt/$max_attempts: Pool $pool_id is in state: $state.
        Waiting..."
        sleep 10
        ((attempt++))
    done

    echo "Timed out waiting for pool $pool_id to be available"
    return 1
}

# Function to wait for a CIDR to be fully provisioned
wait_for_cidr_provisioning() {
    local pool_id=$1
    local cidr=$2
    local max_attempts=30
    local attempt=1
    local state=""

    echo "Waiting for CIDR $cidr to be fully provisioned in pool $pool_id..."

    while [ $attempt -le $max_attempts ]; do
        state=$(aws ec2 get-ipam-pool-cidrs --ipam-pool-id "$pool_id" --query
        "IpamPoolCidrs[?Cidr=='$cidr'].State" --output text)

        if [ "$state" = "provisioned" ]; then
            echo "CIDR $cidr is now fully provisioned (state: $state)"
            return 0
        fi

        echo "Attempt $attempt/$max_attempts: CIDR $cidr is in state: $state.
        Waiting..."
        sleep 10
        ((attempt++))
    done

    echo "Timed out waiting for CIDR $cidr to be provisioned"
    return 1
}

# Step 1: Create an IPAM
echo "Creating IPAM..."
IPAM_RESULT=$(aws ec2 create-ipam \
    --description "My IPAM" \
    --operating-regions RegionName=us-east-1 RegionName=us-west-2)

```



```
if [ $? -ne 0 ]; then
    handle_error "Failed to create IPAM"
fi

IPAM_ID=$(echo "$IPAM_RESULT" | grep -o '"IpamId": "[^"]*' | cut -d'"' -f4)
echo "IPAM created with ID: $IPAM_ID"

# Wait for IPAM to be created and available
echo "Waiting for IPAM to be available..."
sleep 20

# Step 2: Get the IPAM Scope ID - FIXED to correctly identify the private scope
echo "Getting IPAM Scope ID..."
SCOPE_RESULT=$(aws ec2 describe-ipams --ipam-id "$IPAM_ID")

if [ $? -ne 0 ]; then
    handle_error "Failed to get IPAM details"
fi

# Extract the private scope ID directly from the IPAM details
PRIVATE_SCOPE_ID=$(echo "$SCOPE_RESULT" | grep -o '"PrivateDefaultScopeId": "[^"]*' | cut -d'"' -f4)
echo "Private Scope ID: $PRIVATE_SCOPE_ID"

if [ -z "$PRIVATE_SCOPE_ID" ]; then
    handle_error "Failed to get Private Scope ID"
fi

# Step 3: Create a Top-Level IPv4 Pool
echo "Creating Top-level IPv4 Pool..."
TOP_POOL_RESULT=$(aws ec2 create-ipam-pool \
    --ipam-scope-id "$PRIVATE_SCOPE_ID" \
    --address-family ipv4 \
    --description "Top-level pool")

if [ $? -ne 0 ]; then
    handle_error "Failed to create Top-level Pool"
fi

TOP_POOL_ID=$(echo "$TOP_POOL_RESULT" | grep -o '"IpamPoolId": "[^"]*' | cut -d'"' -f4)
echo "Top-level Pool created with ID: $TOP_POOL_ID"
```

```

# Wait for the top-level pool to be available
if ! wait_for_pool "$TOP_POOL_ID"; then
    handle_error "Top-level Pool did not become available in time"
fi

# Provision CIDR to the top-level pool
echo "Provisioning CIDR to Top-level Pool..."
TOP_POOL_CIDR="10.0.0.0/8"
PROVISION_RESULT=$(aws ec2 provision-ipam-pool-cidr \
    --ipam-pool-id "$TOP_POOL_ID" \
    --cidr "$TOP_POOL_CIDR")

if [ $? -ne 0 ]; then
    handle_error "Failed to provision CIDR to Top-level Pool"
fi

echo "$PROVISION_RESULT"

# Wait for the CIDR to be fully provisioned
if ! wait_for_cidr_provisioning "$TOP_POOL_ID" "$TOP_POOL_CIDR"; then
    handle_error "CIDR provisioning to Top-level Pool did not complete in time"
fi

# Step 4: Create a Regional IPv4 Pool
echo "Creating Regional IPv4 Pool..."
REGIONAL_POOL_RESULT=$(aws ec2 create-ipam-pool \
    --ipam-scope-id "$PRIVATE_SCOPE_ID" \
    --source-ipam-pool-id "$TOP_POOL_ID" \
    --locale us-east-1 \
    --address-family ipv4 \
    --description "Regional pool in us-east-1")

if [ $? -ne 0 ]; then
    handle_error "Failed to create Regional Pool"
fi

REGIONAL_POOL_ID=$(echo "$REGIONAL_POOL_RESULT" | grep -o '"IpamPoolId": "[^"]*"'
    | cut -d'"' -f4)
echo "Regional Pool created with ID: $REGIONAL_POOL_ID"

# Wait for the regional pool to be available
if ! wait_for_pool "$REGIONAL_POOL_ID"; then
    handle_error "Regional Pool did not become available in time"
fi

```

```
# Provision CIDR to the regional pool
echo "Provisioning CIDR to Regional Pool..."
REGIONAL_POOL_CIDR="10.0.0.0/16"
PROVISION_RESULT=$(aws ec2 provision-ipam-pool-cidr \
    --ipam-pool-id "$REGIONAL_POOL_ID" \
    --cidr "$REGIONAL_POOL_CIDR")

if [ $? -ne 0 ]; then
    handle_error "Failed to provision CIDR to Regional Pool"
fi

echo "$PROVISION_RESULT"

# Wait for the CIDR to be fully provisioned
if ! wait_for_cidr_provisioning "$REGIONAL_POOL_ID" "$REGIONAL_POOL_CIDR"; then
    handle_error "CIDR provisioning to Regional Pool did not complete in time"
fi

# Step 5: Create a Development IPv4 Pool - FIXED to include locale
echo "Creating Development IPv4 Pool..."
DEV_POOL_RESULT=$(aws ec2 create-ipam-pool \
    --ipam-scope-id "$PRIVATE_SCOPE_ID" \
    --source-ipam-pool-id "$REGIONAL_POOL_ID" \
    --locale us-east-1 \
    --address-family ipv4 \
    --description "Development pool")

if [ $? -ne 0 ]; then
    handle_error "Failed to create Development Pool"
fi

DEV_POOL_ID=$(echo "$DEV_POOL_RESULT" | grep -o '"IpamPoolId": "[^"]*' | cut -d'"' -f4)
echo "Development Pool created with ID: $DEV_POOL_ID"

# Wait for the development pool to be available
if ! wait_for_pool "$DEV_POOL_ID"; then
    handle_error "Development Pool did not become available in time"
fi

# Provision CIDR to the development pool
echo "Provisioning CIDR to Development Pool..."
DEV_POOL_CIDR="10.0.0.0/24"
```

```
PROVISION_RESULT=$(aws ec2 provision-ipam-pool-cidr \
    --ipam-pool-id "$DEV_POOL_ID" \
    --cidr "$DEV_POOL_CIDR")

if [ $? -ne 0 ]; then
    handle_error "Failed to provision CIDR to Development Pool"
fi

echo "$PROVISION_RESULT"

# Wait for the CIDR to be fully provisioned
if ! wait_for_cidr_provisioning "$DEV_POOL_ID" "$DEV_POOL_CIDR"; then
    handle_error "CIDR provisioning to Development Pool did not complete in time"
fi

# Step 6: Create a VPC Using an IPAM Pool CIDR - FIXED to use the correct
# parameter names and a smaller netmask length
echo "Creating VPC using IPAM Pool CIDR..."
VPC_RESULT=$(aws ec2 create-vpc \
    --ipv4-ipam-pool-id "$DEV_POOL_ID" \
    --ipv4-netmask-length 26 \
    --tag-specifications 'ResourceType=vpc,Tags=[{Key=Name,Value=IPAM-VPC}]')

if [ $? -ne 0 ]; then
    handle_error "Failed to create VPC"
fi

VPC_ID=$(echo "$VPC_RESULT" | grep -o '"VpcId": "[^"]*' | cut -d'"' -f4)
echo "VPC created with ID: $VPC_ID"

# Step 7: Verify the IPAM Pool Allocation
echo "Verifying IPAM Pool Allocation..."
ALLOCATION_RESULT=$(aws ec2 get-ipam-pool-allocations \
    --ipam-pool-id "$DEV_POOL_ID")

if [ $? -ne 0 ]; then
    handle_error "Failed to verify IPAM Pool Allocation"
fi

echo "IPAM Pool Allocation verified:"
echo "$ALLOCATION_RESULT" | grep -A 5 "Allocations"

echo ""
echo "IPAM setup completed successfully!"
```

```
echo ""

# Prompt for cleanup
cleanup_resources

echo "Script completed at $(date)"
exit 0
```

- 有关 API 详细信息，请参阅《AWS CLI 命令参考》中的以下主题。

- [CreateIpam](#)
- [CreateIpamPool](#)
- [CreateVpc](#)
- [DeleteIpam](#)
- [DeleteIpamPool](#)
- [DeleteVpc](#)
- [DeprovisionIpamPoolCidr](#)
- [DescribeIpamPools](#)
- [DescribeIpams](#)
- [DescribeVpcs](#)
- [GetIpamPoolAllocations](#)
- [GetIpamPoolCidrs](#)
- [ProvisionIpamPoolCidr](#)

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 SDK 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

您可以使用亚马逊监控亚马逊 EC2 API 请求 CloudWatch，亚马逊会收集原始数据并将其处理为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。这些信息可以让您更好地了解 Web 应用程序的运行状况，并帮助您识别和诊断各种问题。您还可以设置警告，以监控特定阈值，当达到阈值时发送通知或执行特定操作。

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。您必须请求访问此功能。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](#)
- [亚马逊 EC2 API 指标和维度](#)
- [指标数据留存](#)
- [监控代表您发起的请求](#)
- [计费](#)
- [与亚马逊合作 CloudWatch](#)

启用亚马逊 EC2 API 指标

请按照以下过程为您的 AWS 账户请求访问此功能。

请求访问此功能

1. 打开 [AWS Support 中心](#)。
2. 选择创建工单。
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首选联系方式。
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	说明
ClientErrors	客户端错误导致的 API 请求失败次数。 这些错误通常是由于客户端操作引起的，例如在请求中指定了错误或无效的参数，或者以没有权限的用户身份使用某个操作或资源。 单位：个
RequestLimitExceeded	您的账户超出亚马逊 EC2 APIs 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。如果您的请求被节流，会返回 <code>Client.RequestLimitExceeded</code> 错误。 单位：个
AwsTriggeredRequestLimitExceeded	由于操作问题而应用的限制导致 AWS 的 API 请求失败的数量。

指标	说明
	当出现这种限制时，您会收到一条 <code>Client.RequestLimitExceeded</code> 错误消息，表明您的请求因操作问题而受到限制。AWS 这种限制是暂时的，一旦操作问题得到解决，就会自动解决。实现指数退避和重试逻辑，以优雅地处理这些临时限制。 单位：个
<code>ServerErrors</code>	服务器内部错误导致的 API 请求失败次数。 这些错误通常是由 AWS 服务器端错误、异常或故障引起的。 单位：个
<code>SuccessfulCalls</code>	成功完成的 API 请求次数。 单位：个

Dimensions

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](#)。

指标数据留存

亚马逊 EC2 API 指标每隔 1 分钟发送 CloudWatch 一次。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。
- 时间段为 300 秒（5 分钟）的数据点可用 63 天。
- 时间段为 3600 秒（1 小时）的数据点可用 455 天（15 个月）。

监控代表您发起的请求

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。可以使用监视这些请求 CloudWatch。

计费

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

使用以下步骤查看 Amazon EC2 API 指标。

先决条件

您必须允许访问您的账户 EC2 的 Amazon 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，依次选择指标、所有指标。
3. 在浏览选项卡指标下，选择所需的区域以查看指标。
4. 选择EC2指标命名空间。
5. 要查看指标，选择指标维度，例如每实例指标。

Note

指标在两周处于非活动状态后会被隐藏。如果过去两周没有接收到新的数据点，这些指标将不再出现在控制台中，在控制台搜索框中输入指标名称或维度名称时它们也不会显示，并且 [list-metrics](#) AWS CLI 命令也不会返回这些指标。要检索这些指标，请使用[get-metric-data](#)或[get-metric-statistics](#)命令。

使用命令行查看 Amazon EC2 API 指标

使用以下命令之一：

- [list-metrics](#) (AWS CLI)

```
aws cloudwatch list-metrics --namespace "AWS/EC2/API"
```

- [获取-CWMetric 列表](#) (AWS 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。它会根据指标在多个时间段内相对于指定阈值的值，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. 选择选择指标并执行以下操作：
 - 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. 对于时段，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. 对于数值...，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. 对于警报状态触发器，选择警报中。
 - 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. 将显示添加名称和描述页面。
 - a. 对于警报名称，输入警报的名称。名称只能包含 ASCII 字符。
 - b. 对于警报描述，为您的警报输入一个可选的描述。
 - c. 选择下一步。
8. 将显示预览并创建页面。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。